CHAPTER 1
INTRODUCTION

This work gives a tour of the various algorithms in use for the study of the following three important problems.

1) Given a number, determine whether it is prime or composite.
2) Given a number, factor the number completely into its prime factors.
3) Given g, t, p, find a value of x such that $g^x = t \pmod p$

We will first consider first of these three problems.

1.1. DETERMINING WHETHER A GIVEN NUMBER IS PRIME

It is a celebrated theorem of Euler that if GCD(a, p) =1, then $a^{\varphi(p)} \equiv 1 \pmod p$, where φ is the Euler's Phi Function, where φ(p) is the number of numbers less than p that are relatively prime to p. When restricted to primes, φ(p) = p-1 and Euler's theorem becomes the celebrated Fermat's Little Theorem, which states $a^{p-1} \equiv 1 \pmod p$, when a is also relatively prime to p.

To get immediately an idea of whether a given number is prime or composite, we can use the Fermat's Little Theorem. It will give the output in polynomial time, but it is only a probabilistic algorithm. Many composite numbers will pass the test, and they are called pseudo-primes.

In fact, to get more accurate results, more than one base can be tested, but Carmichael Numbers pass the test for any base. An improvement to the Fermat's Little Theorem is the Rabin Miller's Test, in which only fewer composites pass it.

1.2 PRIMALITY PROVING

In order to ensure whether the given number is prime, we have to prove it. Unfortunately, till today there exist no general algorithm to prove easily whether the given number is prime or not. But, for special numbers, there exist classical tests.

In 1876, Lucas used Lucas Sequences to prove that 2^{127}-1 is prime, and this was the largest prime number that stood for 75 years, until in 1951, Miller proved that $(2^{148}+1)/17$ is prime, by using Proth's Theorem. Now this test for Mersenne Numbers has been enhanced by Lehmer, and it is called Lucas Lehmer Test. With the advent of electronic computers over the decades, new record size primes could be discovered, and the largest prime known to date is $2^{43112609}$-1, found out by GIMPS in August 2008.

Similarly, Pépin's Test is used to prove primality of Fermat Numbers. Till date, the largest Fermat Number whose cofactor has been proved to be composite is $2^{2^{27}}+1$. It was realized by Gauss in 1796 at the age of 19 that *a regular polygon of N sides can be constructed by using straightedge and compass if and only if N is a power of 2 times 4 or product of one or more distinct Fermat primes*. Proth's Theorem can be used for Sierpinski Numbers and Lucas Lehmer Riesel for Riesel Numbers.

For general numbers, there are no easy tests, but there are the hard ones. The main such tests are APRT-CLE and Elliptic Curve Primality Proving.

In August 2002, two IIT Kanpur students Kayal and Saxena, and their professor Manindra Agrawal introduced a deterministic polynomial time algorithm for primality proving, that does not lie on any unproved assumptions such as the Riemann Hypothesis. [14] This algorithm is of great theoretical value, since it confirms that PRIMES is in complexity class P, but is of questionable practical value.

Indeed, there do exist polynomial time tests for proving primality for general numbers based upon unproved assumptions. The Riemann Hypothesis is an important unproved conjecture in Analytical Number Theory. It states that all the non-trivial zeros of the Zeta Function in the critical strip $0 < \mathrm{Re}(z) < 1$ lie precisely on the line $\mathrm{Re}(z) = ½$. If proven, many bounds on prime estimates can be improved and primality proving can be simplified.

Assuming that the Extended Riemann Hypothesis is true, if a number N passes the Rabin Miller's Test for all bases from 1 to $2 (\log_2 N)^2$, then N is prime. The Clay Math Institute offers a $1000000 prize for a proof of the Riemann Hypothesis.

1.3 INTEGER FACTORIZATION

Turning to the second problem, namely factoring integers into their prime factors, Trial Division is the most obvious algorithm to factor an integer where we run through all the prime numbers below its square root. For Mersenne Numbers 2^p-1, factors will always be of the form 2kp+1, where p is prime, and for Fermat Numbers 2^{2^p}+1, factors will always be of the form $k.2^{p+2}$+1. But, for general numbers, indeed, there do exist better algorithms.

In 1970, it was barely possible to factor hard 20 digit numbers. Then, in 1974, John Pollard introduced two new algorithms: Pollard's Rho and p-1. These algorithms are used to find small factors, upto 15 digits in size, and are based upon the factor, not the input.

General purpose factoring algorithms are based upon the input. In 1980, Morrison and Brillhart introduced a sub-exponential factoring algorithm, the Continued Fraction Method, by which it was becoming feasible to factor 50 digit numbers. [6] In 1990, Carl Pomerance introduced the Quadratic Sieve, due to which it became feasible to factor 100 digit numbers, with the record being 116 digits.

In 1994, a big team comprising of over 800 people, all over the Internet, factored the 129 digit RSA challenge number, by using the Quadratic Sieve, which had been estimated in Martin Gardner's 1976 Scientific American Column to be safe for 40 quadrillion years. But, now Quadratic Sieve is no longer the champion. It has been replaced by the Number Field Sieve in the spring of 1996, when it was able to factor RSA-130 in only 15% of the time the Quadratic Sieve would have taken. [1]

With the power of computation, large numbers could be factored easily. The record factorization by using the Special Number Field Sieve is 2^{1039}-1 by Kleinjung, et.al. in May 2007. [18] Similarly, for algorithms based upon the factor, the Elliptic Curve Method is the algorithm in use, and it can find out factors upto 70 digits in size. The record is a 73 digit factor from 2^{1181}-1, found out by Joppe Bos, Thorsten Kleinjung, Arjen Lenstra and Peter Montgomery in March 2010.

In 1994, Peter Shor introduced a factoring algorithm for quantum computers, which can factor an integer in polynomial time. In 2001, a couple of workers at IBM factored 15 on a 4-qubit quantum computer by using this algorithm. [15]

Today, a large number of distributed computing projects are based upon the problem of factoring integers. To name a few: Cunningham Project [31], Aliquot Sequence [25], Home Prime Search, Fermat Numbers Factorization, ECMNET, NFSNET, Odd Perfect Number Search, Eleven Smooth, Factoring Smarandache Numbers, Near Repdigits, etc.

The first 11 Fermat Numbers are completely factored. In 1772, Euler showed that 641 is a factor of F_5, which Fermat had previously conjectured that all such numbers are prime. In 1880, Landry found out the factor 274177 from F_6. [11] In 1970, Morrison and Brillhart factored F_7 by using the Continued Fraction Method. [6] In 1980, Brent and Pollard factored F_8 by using a variant of Pollard's Rho. [2] Then, in 1988, F_{11} was the next number to be completely factored, owing to the small size of its penultimate prime factor. [32] In 1990, Lenstra, Lenstra, Manasse and Pollard, along with a large team throughout the Internet, factored F_9 by using the Number Field Sieve. [13] After that, F_{10} was the most wanted number. It was subsequently factored by Brent in 1995 by using the Elliptic Curve Method, with a 40 digit factor. [4]

1.4 DISCRETE LOGARITHM PROBLEM

The third of the problem, called the discrete logarithm problem, comprises of finding out a value of x, given g, t, p such that $g^x = t \pmod p$. It can be done by brute force, or Pollard's Rho, or Pollard's Lambda or Baby Steps Giant Steps. All these have exponential running time, and are described later. A more efficient approach to the discrete logarithm problem is the Pohlig Hellman Algorithm, whose running time depends upon the largest prime factor of $\varphi(p)$. Index calculus method, which comprises of combining smooth relations, has sub-exponential running time.

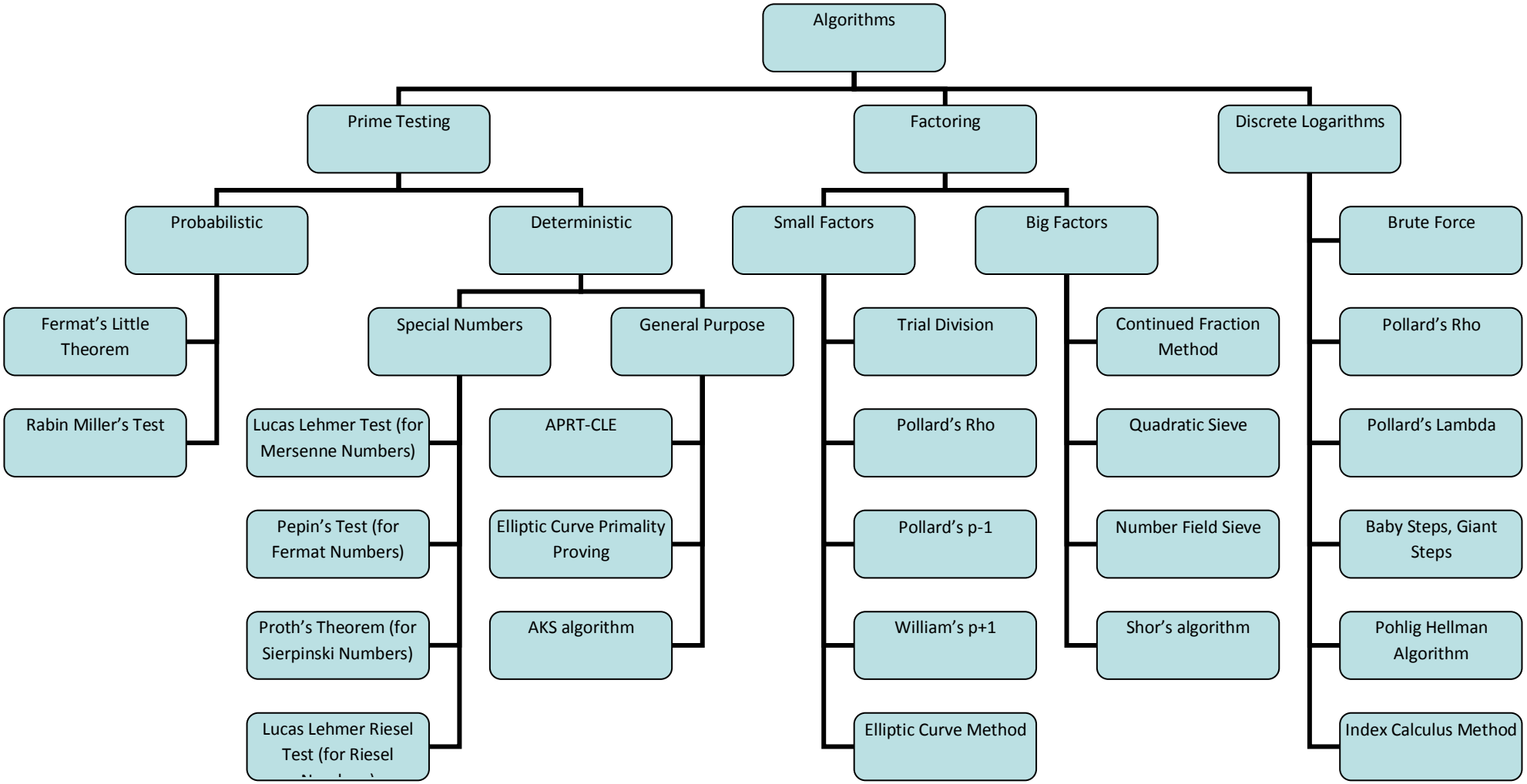
1.5 APPLICATIONS OF PRIME NUMBERS

Prime Numbers do have applications in cryptography, which is the art of sending messages securely over a network. Since a message in its transit is vulnerable to attack by intruders, a message is always encrypted and sent, and it is decrypted by the receiver. An encryption algorithm should be secure such that it is impossible for the intruder for decrypting the message, and read it, with his resources.

Classical algorithms for encryption and decryption are based upon the complexity of factoring integers, with no small factors. In 1974, Diffie-Hellman key exchange was introduced as the first public key cryptosystem. In 1978, Rivest, Shamir and Adleman introduced a new cryptosystem and digital signature algorithm.

We choose two large prime numbers p, q. And compute their product $N = pq$. $\varphi(N) = (p-1)(q-1)$. Let the message be a number $x < N$. Each of the sender and receiver choose their public key as an integer e, relatively prime to $\varphi(N)$. They calculate their private key by using $d = e^{-1} \pmod{\varphi(N)}$. The sender encrypts the message by using receiver's public key, with $y = x^e \pmod{N}$. The receiver decrypts it by using his private key. $x = y^d \pmod{N}$. If the intruder can factor N with his resources, then he can calculate the private key from the public key, and he can decrypt the message. The largest RSA key size to be factored is 768 bits, by Thorsten Kleinjung, et. al. in Bonn University, Germany, in December 2009. This is a 232 digit number, that splits up into two 116 digit prime factors.

Other popular cryptosystems are the El Gamal cryptosystem, which is based upon the Discrete Logarithm Problem, the problem of calculating y, given x and z such that $z = x^y \pmod{p}$, and the Elliptic Curve cryptosystem. Indeed, the Elliptic Curve cryptosystem with a smaller key size is more secure than the RSA cryptosystem of the same key size.



Classification of Algorithms for Integer Factorization, Primality Testing and Discrete Logarithms

CHAPTER 2

INTEGER FACTORIZATION ALGORITHMS

2.1 TRIAL DIVISION

Most obvious algorithm for integer factorization is trial division. To factor an integer N, we test by all the prime numbers till its square root. Indeed, to speed up the computations, we can test by 2, and then by all the subsequent odd numbers.

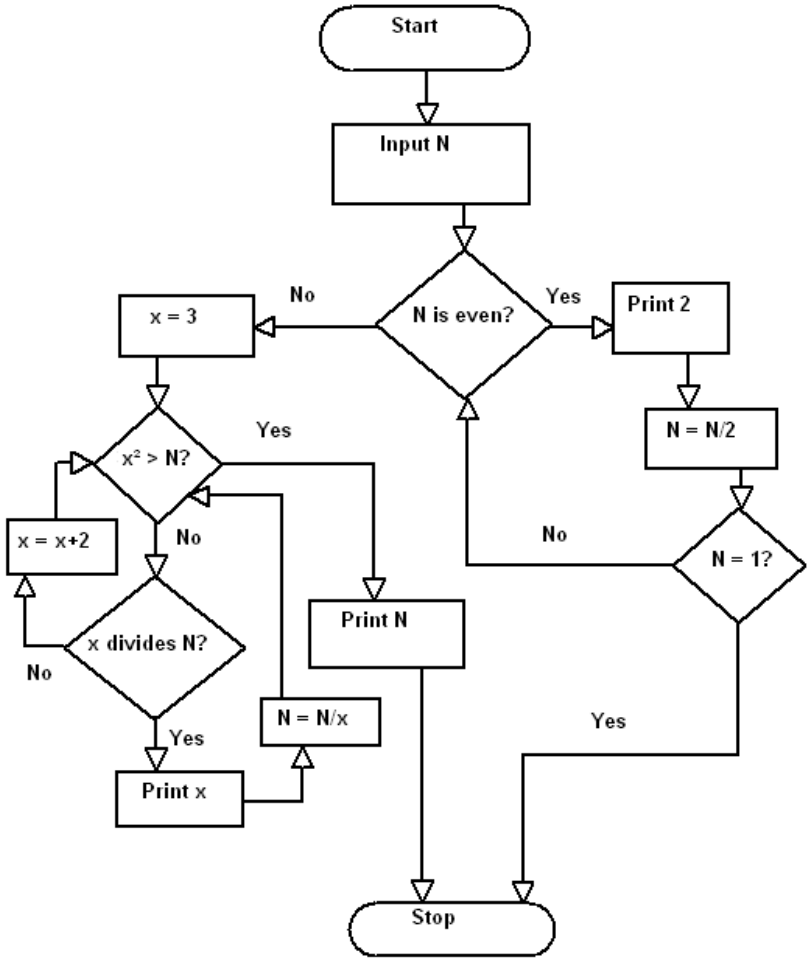


Figure 2.1: A simple integer factorization algorithm

Trial Division can be used to return the small factors, but is not feasible for larger factors. For larger factors, we need to look for better algorithms.

Time Complexity: In the worst case, we have to look for factors upto √N, so the time complexity is Θ(√N).

2.2 POLLARD'S RHO

In 1974, John Pollard introduced a new integer factorization algorithm. Pollard's Rho algorithm is executed on the assumption that Trial Division has eliminated factors upto 10^5 to 10^6. However, in the current form, one run of this algorithm returns only one factor. More factors can be extracted by using repeated runs of the algorithm. No problem that it is not expensive to use multiple runs for returning multiple factors.

In trial division, to factor N, we test by 2, and then examine by all the subsequent odd numbers below √N, in order to find out a prime factor p. That is, in the group {0, 1, 2,…, p-1}, all the factors are examined sequentially, by using a linear function. If a quadratic function is used, say $f(x) = x^2+a \pmod p$, starting with some initial value s in the group, then we need to traverse for only lesser number of values below p, and it will catch in a cycle sooner, and the values (mod p) will repeat.

Since we don't know what p is, we do the calculations (mod N). If two values U and V that are the same (mod p) and different (mod N), then calculating gcd(V-U, N) will return p.

Floyd's cycle finding method: To find a cycle, we don't need to examine all the previous values of U and V. The tortoise hare algorithm (also known as the Floyd's cycle finding algorithm) is an excellent algorithm. First we initialize $U_0 = s$, $V_0 = s$ and then compare U_1 with V_2, U_2 with V_4, U_3 with V_6, etc. U_k with V_{2k}. Thus, each time we increase the number of iterations between U and V by 1. We use the relationship $U_{i+1} = f(U_i)$ and $V_{i+1} = f(V_i)$.

Since computing GCD is expensive when compared to computing successive iterations of U and V, we set aside the task of computing GCD once every 100 iterations

or so, on the assumption that trial division has already eliminated the smaller factors. Pollard's Rho algorithm can easily return prime factors upto 10^{13} or so.

The number of iterations (mod p) needed before the algorithm gets caught in a cycle, is called the Epact of p. To make the algorithm more effective, we need to make the epact lesser, so we need to choose the function carefully. A linear function, ax+b (mod p), $a \neq 0$, will have the Epact to be p. Even quadratic functions such as x^2+a (mod p), will not be efficient, say for a=0. If the function used is x^2-2 (mod p), and x can be written as $y+y^{-1}$ (mod p), then the k^{th} iteration is given by $y^{2^k}+ y^{-2^k}$ (mod p).

Time Complexity: The time complexity of the algorithm depends upon the length of the Epact (mod p), as the other processes do not consume much time. In the set of p values, if a pseudo random function is used, how many iterations do we need on the average, before we catch in a cycle and the values will repeat? The birthday paradox states that out of p values, we need $1.2\sqrt{p}$ values, on the average, such that there is 50% chance that there are atleast two values that are same (mod p). So, it is $\Theta(\sqrt{p})$. For a quadratic function, like x^2+a (mod p), the length of the Epact is conjectured to be $\Theta(\sqrt{p} \ln p)$.The flow chart for this algorithm is given.

Alternate cycle finding algorithms: Richard Brent proposed a new cycle finding method, that will conserve some iterations of U and V, and is estimated to be 24% faster than the tortoise hare algorithm. [3] In the tortoise hare algorithm, we iterate U once and V twice, each time. In Brent's cycle finding algorithm, we save U values for iterations 1, 2, 4, 8, etc., iterations numbered 2^k and for each U value stored, we compare with V for iterations numbered from 2^k+1 to $2^{k+1}-1$.

Such an acceleration was used to factor $F_8 = 2^{2^8}+1$, in 1980, by Brent and Pollard. [2] Since any factor of the Fermat Number F_8 will always be of the form $k.2^{10}+1$, a function such as $x^{2^{10}}+1$ (mod p) will be very effective for this type of factors. On the execution of this algorithm for F_8, the following factor was returned: 1238926361552897. You can use the following mnemonic to remember this factor of F_8: *"I am now entirely persuaded to employ Rho method, a handy trick, on gigantic composite numbers."*

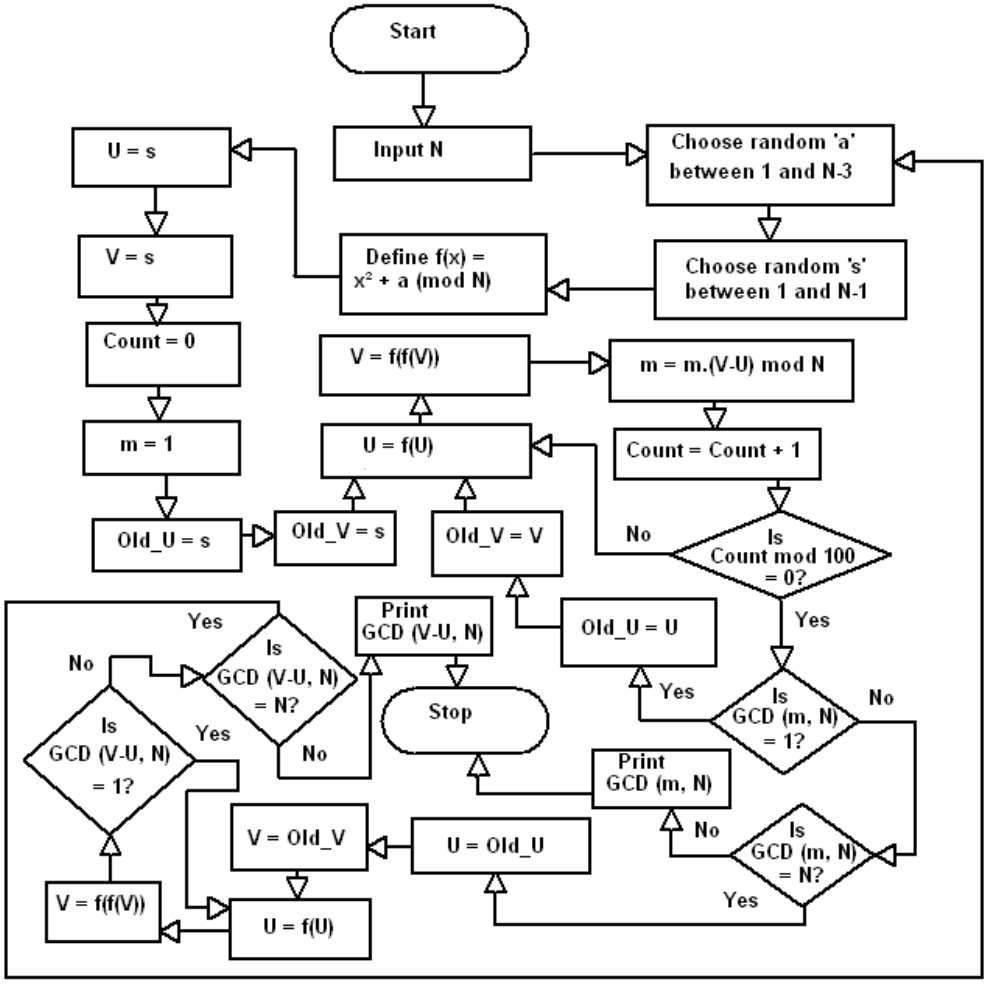


Figure 2.2: Pollard's Rho algorithm

2.3 POLLARD'S p-1 ALGORITHM

In 1974, the second factoring algorithm introduced by John Pollard is the p-1 algorithm. [29] Generally, this algorithm is performed after brute force division and Pollard's Rho have eliminated small factors.

This algorithm gives a factor p, if p-1 is B-smooth, i.e. the largest prime factor of p-1 lies below a bound B. Actually, the basic principle is that, by Fermat's Little Theorem, we have that $a^{p-1} = 1 \pmod p$. Further, $a^{k(p-1)} = 1 \pmod p$, for any multiplier k. So, if we choose a bound B (let R = LCM of all numbers below B) such that p-1|R, then $a^R = 1 \pmod p$. Computing $\gcd(a^R-1, N)$ will return p.

We will perform the algorithm in two stages, with bounds B1 and B2. The algorithm will return a factor p, if all the prime factors of p-1, except the largest lie below B1, and the largest lies below B2.

Stage 1:

Let B1 be the bound for stage 1. Let R be the LCM of all numbers below B1.

$R = q_1^{b_1}.q_2^{b_2}.q_3^{b_3}\ldots q_k^{b_k}$, where q_i are all the primes below B1, and b_i is the largest power of q_i such that $q_i^{b_i} \leq B1$ and $q_i^{b_i+1} > B1$. Let $p = q_1^{a_1}.q_2^{a_2}.q_3^{a_3}\ldots q_k^{a_k} + 1$, where $q_i^{a_i} \leq B1$ for all q_i.

Now, clearly p-1|R, and since $a^{p-1} = 1 \pmod p$ when a, p are relatively prime, so $a^R = 1 \pmod p$. So, $p|GCD(a^R-1, N)$.

The algorithm now proceeds as follows: Let $R = r_1r_2r_3\ldots r_k$, where $r_i = q_i^{b_i}$. Let $a_0 = a$.

Calculate $a_i = a_{i-1}^{r_i} \pmod N$ for all i = 1 to k. We observe that $a_k = a^R \pmod N$, and we evaluate $GCD(a_k-1, N)$. If all the prime factors of p-1 lie below B1, then $GCD(a_k-1, N)$ should return p.

Stage 2:

Suppose that all the prime factors of p-1 except the largest lie below B1 and the largest lie below B2. So, let $p = s.\ q_1^{a_1}.q_2^{a_2}.q_3^{a_3}\ldots q_k^{a_k} + 1$, where p is a prime such that $B1 < s \leq B2$. So, in this case $p|GCD(a_k^s-1, N)$. Let s_j, j = 1 to m, be the set of primes such that $B1 < s \leq B2$. Put $2d_j = s_{j+1} - s_j$, and tabulate it upto the largest prime gap below s_m. Since the largest prime gap below 200000 is only 86, that below 1000000 is only 114, and that below 4.444×10^{12} is only 652, there will not be many values to tabulate. And the big prime gaps occur very rarely. So, there is not a need to tabulate it for all the bigger values.

Now, that we put $b_1 = a_k^{s_1} \pmod N$, and define $b_{j+1} = a_k^{2d_j}b_j \pmod N$ for all j = 2 to m.

Now, we compute $G_t = GCD((b_t-1)(b_{t+1}-1)(b_{t+2}-1)\ldots(b_{t+c-1}-1), N)$ for t = 1, c+1, 2c+1, …, [B2/c]c+1.

Since $b_j = a_k^{s_j} \pmod N$, so we observe that p must divide some G_t.

2.4 WILLIAM'S p+1 ALGORITHM

Analogous to the Pollard's p-1 algorithm, we have a William's p+1 algorithm of integer factorization. This algorithm gives a factor p, if the prime factorization of p+1 is (B1, B2)-smooth, i.e. all the prime factors of p+1 except the largest lie below B1 and the largest lie below B2. So, we need a group of order p+1. For the p-1 factoring algorithm, the order of a (mod p) is a divisor of p-1. A group of order p+1 is based upon Lucas Sequences. [5]

Lucas Sequences:

Let α, β be the roots of the equation $x^2-Px+Q = 0$. We define $U_n = (\alpha^n-\beta^n)/(\alpha-\beta)$ and $V_n = (\alpha^n+\beta^n)$.

When P=1 and Q=-1, then U_n is the Fibonacci Sequence and V_n is the Lucas Sequence.

Further, we have that $U_0 = 0$, $U_1 = 1$, and $U_n = PU_{n-1} – QU_{n-2}$. $V_0 = 2$, $V_1 = P$, $V_n = PV_{n-1} – QV_{n-2}$.

Higher U_n and V_n values can be calculated by using the following formula:

$U_{2n} = V_nU_n$, $U_{2n-1} = U_n^2 – QU_{n-1}^2$, $V_{2n} = V_n^2 – 2Q^n$, $V_{2n-1} = V_nV_{n-1} – PQ^{n-1}$.

Further, that U_n and V_n are divisibility sequences, since $U_n|U_{kn}$ and that $V_n|V_{(2k-1)n}$ and if p is prime then any factor of U_p will always be of the form $2kp\pm1$ and any factor of V_p will always be of the form 2kp+1, when P = 1 and Q = -1. Thus, in this case, that V_p is always congruent to 1 (mod p), and that U_p will always be congruent to ±1 (mod p).

If Δ is given by P^2-4Q, and p is prime, then $U_{p-(\Delta/p)} = 0$ (mod p), so if $(\Delta/p) = -1$, then $U_{p+1} = 0$ (mod p), so the order is p+1. If $(\Delta/p) = 1$, then $U_{p-1} = 0$ (mod p), so the order is p-1. Unfortunately, we don't know the value of p, and the order of group is not always p+1.

So, this algorithm returns a prime factor p, if p+1 is (B1, B2) smooth, and $(\Delta/p) = -1$, or if p-1 is (B1, B2) smooth and $(\Delta/p) = 1$. So, it is a somewhat hybrid (p+1 and p-1) algorithm.

The basic idea is similar to the p-1 algorithm: If $(\Delta/p) = -1$, then $U_{p+1} = 0$ (mod p), further $U_{k(p+1)} = 0$ (mod p) for any multiplier k > 1. So, if we take a value of R that is a multiple of p+1, then $p|GCD(U_R, N)$.

Stage 1:

Let $R = r_1r_2r_3\ldots r_k$ as above, where $r_i = q_i^{b_i}$. So, we need to calculate $a_0 = U_1 = 1$, $a_1 = U_{r_1}$, $a_2 = U_{r_1r_2}$ and so on. For calculating U_{2n} we have a formula, $U_{2n} = V_nU_n = U_n(PU_n - 2QU_{n-1})$. And $U_{2n-1} = U_n^2 - QU_{n-1}^2$. So, from U_n we can calculate U_{2n}, U_{4n}, U_{8n}, etc.

But, there is a major flaw in computing U_{kn} from U_n for the other values of k. For calculating a^{3n}, we can use $a^{2n}.a^n$, for other values of k: we can use modular exponentiation for computing a^{kn}. For example, $a^{59n} = a^{32n}a^{16n}a^{8n}a^{2n}a^n$. But, for the U_n sequence, we <u>cannot</u> compute U_{3n} from U_n and U_{2n} or other values of U_{kn} like this ladder. So, we cannot compute $U_{r_1r_2}$ from U_{r_1} if r_2 is not a power of 2, and $U_{r_1r_2r_3}$ from $U_{r_1r_2}$ if r_3 is not a power of 2, and so on.

The only way to do this computation is to first compute the value of R at first, and then compute U_R in a single step, by using the formula for U_{2n} and U_{2n-1}. The disadvantage is that R can be very large, so for larger bounds B1, it is not practical to store the value of R in a computer. Also, that if the bound B1 is increased to a new bound B1', then we have to compute the value of $U_{R'}$ starting from the scratch, starting from U_1 and U_2, instead of continuing it from U_R. So, we have to look for a different technique.

Let us assume that $P' = (\alpha/\beta)+(\beta/\alpha)$ and $Q' = (\alpha/\beta).(\beta/\alpha) = 1$. Let us denote $U_n(P,Q)$ and $V_n(P,Q)$ as the sequence generated by the roots of the equation $x^2-Px+Q = 0$. We have $U_{2m}(P,Q) = PQ^{m-1}U_m(P',1) \pmod N$. So, if $p|U_R(P,Q)$ then $p|U_{2R}(P,Q)$ and so $p|U_R(P',1)$. So, we lose no generality in assuming that $Q = 1$. We will assume throughout that $Q = 1$.

If p is odd prime and p does not divide Q, then we have that $U_{[p-(\Delta/p)]m} = 0 \pmod p$, and $V_{[p-(\Delta/p)]m} = 2Q^{m[1-(\Delta/p)]/2} \pmod p$. So, if $Q = 1$, then $V_{[p-(\Delta/p)]m} = 2 \pmod p$.
So, if $p|U_R(P,1)$ then $p|(V_R(P,1) - 2)$. So, we can go up the ladder by using the V_n sequence.

So, we need to compute $\gcd(V_R - 2, N)$, where $R = r_1r_2r_3\ldots r_k$. We will always use $Q = 1$.

We have $V_{nk}(P,Q) = V_n(V_k(P,Q),Q^k)$, so with Q = 1, it becomes $V_{nk}(P,1) = V_n(V_k(P,1),1)$. So, for the stage 1 of the algorithm, we use a starting value P_0 such that $\gcd(P_0^2 - 4, N) = 1$
We calculate $P_1 = V_{r_1}(P_0,1)$, $P_2 = V_{r_1r_2}(P_0,1) = V_{r_2}(V_{r_1}(P_0,1),1) = V_{r_2}(P_1,1)$,
$P_3 = V_{r_1r_2r_3}(P_0,1) = V_{r_3}(V_{r_1r_2}(P_0,1),1) = V_{r_3}(P_2,1)$ and similarly we calculate $P_m = V_{r_m}(P_{m-1},1) \pmod N$ for all m = 1 to k. We observe that $P_k = V_R(P_0,1)$.

To calculate $V_r(P,1)$, we use the formula: $V_0 = 2$, $V_1 = P$, $V_{2r-1} = V_rV_{r-1} - P \pmod N$, $V_{2r} = V_r^2 - 2 \pmod N$. We then compute $\gcd(P_k - 2, N)$ to check if stage 1 returns any factor.

Stage 2:

Suppose that if $(\Delta/p) = -1$ and the largest prime factor of p+1 be s. Let s be a prime such that $B1 < s \leq B2$. Let all other factors of p+1 lie below B1. Then, $p|(V_s(P_k,1) - 2)$ and also $p|U_s(P_k,1)$.

We tabulate U_{2d_j-1}, U_{2d_j}, U_{2d_j+1} for all the distinct d_j till the largest gap of successive primes between B1 and B2, by using $U_0 = 0$, $U_1 = 1$, $U_{n+1} = P_kU_n - U_{n-1}$.
Further, we have that $U_{nk}(P,Q) = U_n(V_k(P,Q),Q^k).U_k(P,Q)$, and $U_{m+n} = U_mU_{n+1} - QU_{m-1}U_n$, $\Delta U_{m+n} = V_mV_{n+1} - QV_{m-1}V_n$.
So, we put that $T[s_i] = \Delta U_{s_i}(P_k,1) = \Delta U_{s_iR}(P_0,1)/U_R(P_0,1) \pmod N$.
Substituting this in the above equation, we can get that
For s_1, we can use $T[s_1] = P_kV_{s_1}(P_k,1) - 2V_{s_1-1}(P_k,1) \pmod N$ and $T[s_1\text{-}1] = 2V_{s_1}(P_k,1) - P_kV_{s_1-1}(P_k,1) \pmod N$. For other s_j, $2 \leq j \leq m$, we can use $T[s_{j+1}] = T[s_j]U_{2d_j+1}(P_k\text{-}1) - T[s_j\text{-}1]U_{2d_j}(P_k\text{-}1) \pmod N$, and $T[s_{j+1}\text{-}1] = T[s_j]U_{2d_j}(P_k\text{-}1) - T[s_j\text{-}1]U_{2d_j-1}(P_k\text{-}1) \pmod N$.
We compute $H_t = \text{GCD}(T[s_t].T[s_{t+1}].T[s_{t+2}]\ldots T[s_{t+c-1}], N)$ for all t = 1, c+1, 2c+1, …, [B2/c]c+1. Since $p|U_s(P_k,1)$ if $(\Delta/p) = -1$, so p must divide some H_t.

Special Applications:

Let N = p.q where p and q are the prime factors of N. p+1 algorithm will find out a factor p, in which p-1 algorithm has failed because it is *degenerate*. That is, the largest prime factor of both p-1 and q-1 is the same. In this case, p-1 algorithm will return N, neither p, nor q, and will fail. For example consider N =

676738418869182417666253239653933743825490524168317486197006720422834276095213379928417101517466403166855582327009727764197997662033204334608729422523991620653 01

But, p+1 algorithm, if Δ is chosen such that (Δ/p) = 1 and (Δ/q) = -1, with the same bound as that was used in the p-1 algorithm, since p-1 is smooth, and q-1 is smooth but not q+1, the p+1 algorithm will return the factor p.

On the other hand, if we choose Δ such that (Δ/p) = -1 and (Δ/q) = 1, since p-1 and q-1 are smooth, and not p+1, the algorithm will return the factor q.

2.5 ELLIPTIC CURVE METHOD

Lenstra's elliptic curve factorization algorithm is an extension to Pollard's p-1 algorithm. The p-1 algorithm returns a factor p from N, if p-1 is (B1, B2) smooth. William's p+1 algorithm returns a factor p if either, p-1 or p+1 is smooth, depending upon the coefficients of the recurrence equation. By the use of elliptic curves, we allow a larger range of values for the group order, which if smooth, depending upon the curve, will return the factor p. [22]

Elliptic Curves over Finite Fields

Elliptic Curves are defined by using the equation

$y^2 = x^3 + Cx^2 + Ax + B \pmod p$

This is the affine form of the elliptic curve, in which one point is assigned to the point at infinity, O. This is the identity element of the group. The affine form can be derived from the projective form of the elliptic curve $y^2z = x^3 + Cx^2z + Axz^2 + Bz^3 \pmod p$ in which the point (x, y, z) of the projective form corresponds to the point (x/z, y/z) of the affine form. The point at infinity of the projective form is given by (0, 1, 0). Note that in the

projective form of the elliptic curve, (x, y, z) is a point on it if and only if (tx, ty, tz) is also a point on it.

Elliptic Curve Addition

Consider the affine form of the elliptic curve. We can utilize the tremendous power of elliptic curves only if we define some addition operation, that makes the set of points on the elliptic curve to be an Abelian Group. We define the addition operation as follows:

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ and O be the point at infinity. Then,

$-O = O$

$-P_1 = (x_1, -y_1)$

$O + P_1 = P_1$

If $P_2 = -P_1$, then $P_1 + P_2 = O$

If $P_2 \neq -P_1$, then $P_1 + P_2 = (x_3, y_3)$, where

$x_3 = m^2 – C – x_1 – x_2$

$-y_3 = m(x_3 – x_1) + y_1$

Where the slope m is given by

$m = (y_2 – y_1)/(x_2 – x_1)$ if $x_2 \neq x_1$

$m = (3x_1^2 + 2Cx_1 + A)/(2y_1)$ if $x_2 = x_1$

Group Order of an Elliptic Curve

We define the group order #E of the elliptic curve to be the number of points in the elliptic curve. Since the set of points in the elliptic curve form an Abelian Group under addition, we have [#E]P = O, where [k]P means that the point P is multiplied k times. The factor p is returned if #E is (B1, B2) smooth. The group order of the elliptic curve lies between $p+1-2\sqrt{p}$ and $p+1+2\sqrt{p}$.

It is the Z coordinate that determines the point at infinity, since for the point at infinity, the Z coordinate is 0. We succeed if [#E]P = O (mod p) and not [#E]P = O (mod N). We then calculate p = gcd(Z, N).

So, we use a form of the elliptic curve that makes use of the Z coordinate.

Montgomery form

The above Weierstrass form of the Elliptic Curve is not the best in terms of computations. The Montgomery form is used instead.

$gy^2 = x^3 + Cx^2 + x \pmod p$, where A = 1 and B = 0.

For the Montgomery form, the group order is always a multiple of 4.

The Montgomery form makes use of X and Z coordinates, (X:Z) and does not matter about the Y coordinate.

Suyama discovered a parameterization for which the group order is always a multiple of 12, thereby increasing the probability that it is smooth.

Take a random value of $\sigma \neq -1, 0, 1, 5$

Define $u = \sigma^2 - 5$, $v = 4\sigma$, $C = [(v-u)^3(3u+v)/4u^3v] - 2$

Then the group order will always be a multiple of 12.

We define the starting point (X:Z) to be $(u^3:v^3) \bmod N$.

For Montgomery coordinates, we have to perform addition operation without the use of Y coordinates. Clearly, for any given X coordinate, there will be 2 distinct values of Y coordinate such that their sum is p. Let $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$. We define X_+ as the X coordinate of $P_1 + P_2$, X_- as the X coordinate of $P_1 - P_2$, Z_+ as the Z coordinate of $P_1 + P_2$, Z_- as the Z coordinate of $P_1 - P_2$.

Then we can calculate them as follows:

$(X_+, Z_+) = addh(X_1, Z_1, X_2, Z_2, X_-, Z_-)$

$X_+ = Z_-((X_1X_2 - AZ_1Z_2)^2 - 4B(X_1Z_2 + X_2Z_1 + CZ_1Z_2)Z_1Z_2)$ and

$Z_+ = X_-(X_1Z_2 - X_2Z_1)^2$

Note that for Suyama's parameterization, we have A=1 and B=0. So, we will have lesser number of field multiplications for point addition, and the elliptic curve addition operation to be homogeneous.

For doubling a point, let X_+ and Z_+ be the X and Z coordinates of $[2]P_1$. Then, we have that $(X_+, Z_+) = doubleh(X_1, Z_1)$

$X_+ = (X_1^2 – AZ_1^2)^2 – 4B(2X_1 + CZ_1)Z_1^3$, and

$Z_+ = 4Z_1(X_1^3 + CX_1^2Z_1 + AX_1Z_1^2 + BZ_1^3)$.

Elliptic Curve Multiplication

For multiplying any point P, with a scalar k, to calculate [k]P, we proceed as follows:

For example, to calculate [7]P from P, we calculate [2]P = doubleh(P),

[3]P = addh([2]P, P, P)

[4]P = doubleh([2]P)

[7]P = addh([4]P, [3]P, P)

The following algorithm is used to multiply any point P (X:Z) by a scalar k.

If k = 0, then return O

If k = 1, then return (X:Z)

If k = 2, then return doubleh(X:Z)

If k > 2, then we define as follows:

{

(U:V) = (X:Z)

(T:W) = doubleh(X:Z)

Let k be represented as a B-bit integer in binary notation.

$k_{B-1}k_{B-2}k_{B-3}\ldots k_1k_0$

for j = B-2 down to 0

{

if (k_j = 1) then

{

(U:V) = addh(T:W, U:V, X:Z)

(T:W) = doubleh(T:W)

}

Else if (k_j = 0) then

{

(T:W) = addh(U:V, T:W, X:Z)

(U:V) = doubleh(U:V)

}

}

Return (U:V)

}

Elliptic Curve Factoring Algorithm

We perform the algorithm in two stages, with bounds B1 and B2. This algorithm returns a factor p, if all the factors of #E except the largest lie below B1 and the largest lies below B2.

Stage 1

In stage 1, for every prime p below B1, we take the largest power of that prime below B1 and multiply it by the resulting point, starting with the initial point $(u^3:v^3)$ for the prime p = 2. Let the Q be the resulting point after all primes below B1 have been executed. At the end of stage 1, we take the GCD of the resulting Z coordinate with N, and check if it is not 1 or N. If so, we return the factor p, and stop. If the GCD is 1, we proceed on to stage 2, if the GCD is N, we try out with a smaller B1 value for the same curve.

Stage 2

In stage 2, starting with the initial point Q, we multiply it by the first prime after B1 and subsequently add it by the product of Q with the difference between successive primes between B1 and B2. We then take the GCD of the Z coordinate of the resulting point, with N, every 1000 iterations or so, depending upon the value of D. The following algorithm illustrates the stage 2 of the algorithm:

S_1 = doubleh(Q_1)

S_2 = doubleh(S_1)

For d from 3 to D

{

S_d = addh(S_{d-1}, S_1, S_{d-2})

β_d = X(S_d) * Z(S_d) mod N

}

g=1

B=B1-1, if B1 is even; B should always be odd

T=[B-2D]Q

R=[B]Q

For r = B to B2; r = r+2D

{

$\alpha = X(R) * Z(R) \bmod N$

For each prime q between r+2 and r+2D

{

$\delta = (q-r)/2$

$g = g((X(R) - X(S_\delta))(Z(R) + Z(S_\delta)) - \alpha + \beta_d) \bmod N$

}

$(R, T) = (addh(R, S_D, T), R)$

}

g = gcd(g, N)

If $1 < g < N$, then return g

Else, change the σ value, and try some other curve.

The following table gives the number of curves and the optimal values of B1 and B2 needed to uncover an M-digit factor p, from N.

Digits	B1	B2	Curves
20	11000	1873422	77
25	50000	12746592	206
30	250000	128992510	401
35	1000000	1045563762	948
40	3000000	5706890290	2440
45	11000000	35133391030	4590
50	43000000	240490660426	7771
55	110000000	776278396540	17899
60	260000000	3178559884516	43670
65	850000000	15892628251516	69351

Table 2.1: Optimal B1 and B2, number of curves for ECM

For example, if you want to find out factors upto 20 digits long, then you can try 77 curves with B1 = 11000 and B2 = 1873422. If it fails, the probability that there remains a factor of 20 digits is about 1/e. If you perform 154 curves, with the same B1 and B2 values, the probability to miss a 20 digit factor is about $1/e^2$, and so on. So, you can increase the B1 and B2 values to B1 = 50000 and B2 = 12746592, and then thus perform 206 curves with these B1 and B2 values, and so on.

CHAPTER 3
GENERAL PURPOSE FACTORING ALGORITHMS

General Purpose Factoring Algorithms are based upon Fermat's Factorization Method.

3.1 FERMAT'S FACTORIZATION METHOD

Fermat's Factorization Method is the process of factoring a number by expressing it as the difference of two squares. Since every composite number can be expressed as the difference of two squares, it can subsequently be factored. If a composite number is the product of two numbers N = pq, then it can be written as $(a+b)(a-b) = a^2 - b^2$, where $a = (p+q)/2$ and $b = (p-q)/2$.

The process is to find two numbers a and b such that $a^2 \equiv b^2 \pmod{N}$. Once, we have such a congruence, we can calculate GCD(a+b,N) and GCD(a-b,N) to get its factors.

A Contest Problem

If someone in a contest asks to factor 8051, in this case it is useless to trial divide through all the primes starting from 2. A quick observation will be, since $8051 = 8100 - 49 = 90^2 - 7^2 = (90+7)(90-7) = 83 \times 97$.

By brute force division, we would traverse forward by dividing by all the primes starting with 2. In Fermat's Method, one would start at $\sqrt{N}$, and traverse backwards until $a^2 - N$ is a perfect square. Since, a general number chosen at random, has more chances to have smaller factors, Fermat's Method is essentially worthless for a random number. It is only generally suited for smaller numbers.

Certainly there are many enhancements to the Fermat's Method. One such enhancement is the elimination of certain residue classes in which the factors cannot lie. For example,

if $N \equiv 1 \pmod 4$, then a cannot be even. If $N \equiv 2 \pmod 3$, then a must be a multiple of 3. If $a^2 - N = 17$, then $a \equiv \pm 1, \pm 2, \pm 4, \pm 8 \pmod{17}$.

3.1.1 LEHMAN'S METHOD

Another enhancement to Fermat's Method is the search for a multiplier. We want to search for $a^2 \equiv b^2 \pmod{kN}$, and subsequently, we can calculate gcd(a+b , N) and gcd(a-b , N). If N = 2581, then by Fermat's Method, we have to go till a = 59, starting with $\sqrt{N} = a = 51$, we get $59^2 - 2581 = 900 = 30^2$. But, if we use a multiplier k = 3, then kN = 7743, and we have immediately $88^2 - 1^2 = 89 \times 87$. Hence, $2581 = 89 \times 29$.

The following algorithm of Lehman formalizes the search for a multiplier.

If N has a non-trivial divisor $d \leq N^{1/3}$, then return d.
For $1 \leq k \leq n^{1/3}$ {
For $2\sqrt{(kn)} \leq a \leq 2\sqrt{(kn)} + n^{1/6}/(4\sqrt{k})$ {
If $b = \sqrt{(a^2 - 4kn)}$ is an integer, return gcd(a + b, n);
}
}
Return "N is prime".

3.2 DIXON'S FACTORIZATION METHOD

It is based upon the Fermat's Factorization Method. For larger numbers, it will be very difficult to find out a congruence such that $a^2 \equiv b^2 \pmod N$. So, the idea is to find out many squares (mod N), whose product is a square. For example, let us take N = 1649. $41^2 - 1649 = 32$, $42^2 - 1649 = 115$, $43^2 - 1649 = 200$, none of them is a perfect square. But their product $(41.43)^2 \bmod 1649 = 32 \times 200 = 6400$ form a perfect square. So we have $114^2 \equiv 80^2 \pmod{1649}$. So its factors are gcd(114-80,1649) = 17 and gcd(114+80,1649) = 97. So, we have that $1649 = 17 \times 97$.

The technique is to choose a collection of small primes, called the Factor Base, and then find out values of $a^2 \pmod N$ whose all prime factors lie within the Factor Base primes, i.e. they are smooth to the factor base, by using random values of a. Each such

congruence is called a relation, and our aim is to find out many relations whose product is a perfect square. Since values of a^2 (mod N) are more likely to be smooth near $\sqrt{kN}$, for integral values of k, such values are tried out.

For example, let N=1111111 and factor base limit = 20.

We need to find values of $a^2 - N$, whose all prime factors lie below 20.

For example, $1066^2 - 1111111 = 3^3 * 5 * 11 * 17$

$1067^2 - 1111111 = 2 * 3^4 * 13^2$

$1093^2 - 1111111 = 2 * 3^3 * 7 * 13 * 17$

$1121^2 - 1111111 = 2 * 3^3 * 5 * 7^2 * 11$

$1134^2 - 1111111 = 5 * 11^2 * 17^2$

$1156^2 - 1111111 = 3^2 * 5^2 * 7 * 11 * 13$

Sometimes, as the value of N grows in size, it may be very difficult to get all the relations smooth within the factor base primes. So, we also allow partial relations, which contains 1 or more large primes that are slightly greater than the factor base primes. Once two relations or more have been accumulated, that contains the same large prime, we can merge them in order to get a relation which is smooth within the factor base primes.

So, to find out a subset of these smooth relations whose product is a perfect square, we build a matrix with the relations as rows and the factor base primes as columns. For the factor base, only those primes p, such that the Legendre Symbol (kN/p) = 1 are sufficient. So, different values of k generate different factor bases. The matrix entries contain only 0 and 1, and for every prime p, whose power is odd, we associate a 0 and if the power is even, we associate a 1 with it.

a	**2**	**3**	**5**	**7**	**11**	**13**	**17**
1066	0	1	1	0	1	0	1
1067	1	0	0	0	0	0	0
1093	1	1	0	1	0	1	1
1121	1	1	1	1	1	0	0
1134	0	0	1	0	0	0	0
1156	0	0	0	1	1	1	0

So, to find a subset whose product is a perfect square, we must find a linearly dependent subset of rows, i.e. a set of vectors which add up to the zero vector in Z_2. For finding such a subset, algorithms such as Gaussian Elimination are available, which finds out a linear dependent set of vectors, by bringing the matrix to an upper triangular matrix. But, Block Lanczos algorithm [16] is faster for larger and sparse matrices. Block Weidemann is similar to Block Lanczos algorithm, but it is suitable for parallelizing. By elementary linear algebra, there needs to be at least M+1 relations if the factor base contains M primes, in order to guarantee that there is a linear dependency.

For example, relations {1, 2, 3, 5, 6} form a linearly dependent set of vectors. So, we have that $(1066^2)(1067^2)(1093^2)(1134^2)(1156^2) = 2^2 * 3^{12} * 5^4 * 7^2 * 11^4 * 13^4 * 17^4$ (mod 1111111). So, $62280^2 = 114382^2$ (mod 1111111)

Once such a congruence has been established, there is 50% chance that it will give rise to non-trivial factorization, that is not 1 and N, or any multiple thereof.

From the above dependency, the factors are

gcd(114382-62280, 1111111) = 239 and gcd(114382+62280, 1111111) = 4649

3.3 CONTINUED FRACTION METHOD

In the Dixon's Factoring Method, we need to find many relations, values of x^2-kN, that are smooth to the factor base, so that we can find out a subset whose product is a perfect square. Taking random relations, and trial dividing each relation by the factor base primes is time consuming, and it is difficult to get even a single relation for larger numbers. So, we need to find out a systematic method of finding out such relations.

Take the continued fraction expansion of $\sqrt{kN} = q_0 + 1/(q_1 + 1/(q_2 + 1/(q_3 + \dots)))$

Let the convergents of $\sqrt{kN}$ be given by A_n/B_n.

The convergents can be calculated by using the following recurrence.

We have $A_0 = q_0$, $B_0 = 1$, $A_1 = q_0q_1+1$, $B_1 = q_1$

$A_i = q_iA_{i-1} + A_{i-2}$, $B_i = q_iB_{i-1} + B_{i-2}$ for all $i \geq 2$.

Each of the pairs of values A_n and B_n satisfy the congruence

$A_{n-1}^2 - kNB_{n-1}^2 = (-1)^n Q_n$, where $q_n = (\sqrt{kN} + P_n)/Q_n$.

So, we have $A_{n-1}^2 \equiv (-1)^n Q_n \pmod N$, which gives a square on the left hand side, and the right hand side is a value, which has to be expressed as a product of primes.

Further, we always have $0 \leq P_n < \sqrt{kN}$ and $0 < Q_n < 2\sqrt{kN}$, so that the Q_n values are much smaller than N, and they have high probability to be smooth.

The A_{n-1} and Q_n values can be quickly calculated by using the following formula:

Initially, we take $A_{-2} = 0$, $A_{-1} = 1$, $Q_{-1} = kN$, $r_{-1} = g$, $P_0 = 0$, $Q_0 = 1$, $g = [\sqrt{kN}]$

And then iterate by using the following set of formula. We factor every Q_n value to determine if it is smooth. Note that we do not make use of B values.

$g + P_n = q_n Q_n + r_n$, where $0 \leq r_n < Q_n$

$A_n \equiv q_n A_{n-1} + A_{n-2} \pmod N$

$g + P_{n+1} = 2g - r_n$

$Q_{n+1} = Q_{n-1} + q_n(r_n - r_{n-1})$

Linear Algebra

So, we build a matrix with the relations as rows and the factor base primes as columns. Along with the matrix, we associate a history matrix, which is a square matrix with number of rows equal to the number of relations accumulated. Initially, the history matrix contains 1 only in its diagonal elements. While performing Gaussian Elimination, while adding a row to another row, the history matrix is also correspondingly adjusted. After performing Gaussian Elimination, once a zero vector is spotted out, the corresponding row of the history vector is analyzed to pick out the subset of relations which when multiplied together, their product is a perfect square on the right hand side.

Square Root

Once the subset of relations is obtained, the square root procedure gives the square root of the product of Q_i's which is a perfect square. Let there be s relations Q_i are

the various relations, this procedure calculates Q', the least positive remainder of square root of product of Q_i's (mod N).

(1) I = 2
(2) Q' = 1
(3) R = Q_1
(4) X = GCD (R, Q_I)
(5) Q' = XQ' (mod N)
(6) R = (R/X)(Q_I/X)
(7) I = I + 1
(8) If I ≤ s, then goto (4)
(9) X = √R
(10) Q' = XQ' (mod N)

Now, we have that the product of various A_{i-1}'s ≡ Q' (mod N), which helps to establish a congruence of squares.

Records

In 1970, Morrison and Brillhart factored the seventh Fermat number $F_7 = 2^{2^7} + 1$, by using the continued fraction method. [6] The factor is found out to be: 59649589127497217. Use the following mnemonic to remember the factor:
First exhibitor clever John Brillhart, after numerous Saturdays, "I am allowed only primality testing on a weekday."

3.4 QUADRATIC SIEVE

The working principle of quadratic sieve is very similar to Dixon's factorization method, in which we find many smooth values of x^2 – kN, which we merge together in order to get their product to be perfect square. The working steps are the same, first we find many smooth relations, and then solve the matrix to get the linear dependency, and then find the square root. The last two steps are precisely the same, there is no difference

between them. The Dixon's Method, Continued Fraction Method, and the Quadratic Sieve vary only in the way how the relations are accumulated.

In the Dixon's Factorization Method, we choose random values of x, and check if they are smooth. In this way, it is very difficult to get any smooth relation as N grows larger. So, in the Continued Fraction Method, we designed a systematic method of accumulating relations, which come from the continued fraction expansion of $\sqrt{kN}$. For any convergent A_n/B_n, we have $A_n^2 - kNB_n^2 = (-1)^n Q_n$, so we have that $A_n^2 \equiv (-1)^2 Q_n$, which accumulates as a relation if Q_n is smooth. Further, since $0 \leq Q_n < 2\sqrt{kN}$, there is a greater chance for the Q_n values to be smooth.

However, in the continued fraction method, we have to trial divide every value of Q_n by the factor base primes, to check if it is smooth. This is a very time consuming process, but we have better methods. In Quadratic Sieve, we can use a sieve to determine those values of x, for which any factor base prime divides. The values of such x, will follow a linear pattern, so it is very fast.

Suppose that a prime p, in the Factor Base, divides $x^2 - kN$, so we will have $x^2 - kN \equiv 0 \pmod{p}$, or $x^2 \equiv kN \pmod{p}$. So, the values of x for which the prime p divides $x^2 - kN$ are given by $\pm\sqrt{kN} \pmod{p}$. This is a modular square root (mod p), and so it will have either two solutions or no solutions (mod p), corresponding to which kN is a quadratic residue or a quadratic non-residue (mod p). However, if kN is divisible by p, then the two solutions $x^2 \equiv 0 \pmod{p}$ are coincident. Note that if there are two solutions (mod p), they are complement of each other, that is, they add up to p.

In case there are no solutions (mod p), because kN is a quadratic non-residue (mod p), then the prime p divides no value of $x^2 - kN$. So, we need not have this prime p in the factor base. So, for choosing the factor base primes, only those primes p, for which the Legendre Symbol $(kN/p) = 1$, are sufficient.

However, in case there are two solutions (mod p), they are complements of each other, and they are given by $x \equiv \pm\sqrt{kN} \pmod{p}$. These are the two residue classes of x for which the prime p divides $x^2 - kN$. We can sieve by every prime in the factor base, in these two residue classes, in which two adjacent values of x, in the same residue class

(mod p), have a common difference of p. The advantage of sieve is that we can only test at regular intervals for which a prime p divides $x^2 - kN$, instead of testing every value of x, if the prime p divides $x^2 - kN$. However, if $kN \equiv 0 \pmod{p}$, then the two square roots are coincident, and $x \equiv 0 \pmod{p}$ is the only residue class, for which we have to sieve for the values of x, for the prime p.

Within these steps, we have various enhancements. Instead of starting at $x = \sqrt{kN}$ and progressing ahead, we could have various values of x, centered at $\sqrt{kN}$. However, it allows the values of $x^2 - kN$, which are negative, but it can be managed by including -1 in the factor base, only one more additional element in the factor base. In case the value of $x^2 - kN$ is negative, then the matrix entry corresponding to -1 and this relation is 1, since the power is assumed to be odd, and if the value of $x^2 - kN$ is positive, then the matrix entry is 0, since the power is even. While merging the rows of the matrix while solving it, to get a zero vector, the sum corresponding to -1 element should also be 0, or even, so that the product of relations is a perfect square, which is positive.

Also that, in order for the execution of the algorithm to be fast, for every prime p dividing $x^2 - kN$, we need not store the value of $x^2 - kN$, and keep on dividing it by every prime p, as it divides $x^2 - kN$. What we do is that, for every block of values of x, we use a floating point variable for every value of x, starting from 0, and at regular intervals of p, of the two residue classes, which we sieve over, as the prime p divides the value of $x^2 - kN$, we add the logarithm of the prime p, to the floating point variable. And at the end, if the factor base bound is FB and the large prime bound is UB, then the smooth relations are given by those values of x, whose floating point variables, have accumulated more than the value of $\log(x^2 - kN) - \log(UB)$.

Hensel's Lifting can be used to sieve by the prime powers. Suppose that r is a root of the equation $f(x) \equiv 0 \pmod{p^{m-1}}$, and that $f(r) \equiv 0 \pmod{p^{m-1}}$. Then we can calculate the roots of f(x) mod p^m, by using the Hensel's Lemma. Let $t = [-(f(r))/p^{m-1}].(f'(r))^{-1} \pmod{p}$. Then, we have $f(r + tp^{m-1}) \equiv 0 \pmod{p^m}$. If x = q is a root of $f(x) \equiv 0 \pmod{p^m}$, then $x = p^m - q$ is the other root of this equation. Note that if $f(r) \equiv 0 \pmod{p^{m-1}}$ and $f'(r) \neq 0 \pmod{p}$, then $f(r) \equiv 0 \pmod{p^m}$ has a unique solution as given above. However if $f(r) \equiv 0 \pmod{}$

p^{m-1}) and f'(r) ≡ 0 (mod p), and in addition f(r) ≡ 0 (mod p^m), then f(r + tp^{m-1}) ≡ 0 (mod p^m) for all integers t. But, if f(r) ≡ 0 (mod p^{m-1}) and f'(r) ≡ 0 (mod p), and f(r) ≠ 0 (mod p^m), then f(x) ≡ 0 (mod p^m) has no solution for any x ≡ r (mod p^{m-1}).

Note that we use the polynomial $x^2 - kN$, and so we get $t = [(kN - r^2)/p^{m-1}].(2r)^{-1}$ (mod p). But for the prime p = 2, we have to sieve separately. Since any odd square is congruent to 1 (mod 8), so if kN ≡ 3 (mod 4), then only 2 divides all odd values of x for $x^2 - kN$. If kN ≡ 5 (mod 8), then only 4 divides all odd values of x for $x^2 - kN$. If kN ≡ 1 (mod 8), then 8 divides all odd values of x for $x^2 - kN$. Sieving by higher powers of 2 can be done by using the fact that there are 2 values of x less than 2^{n-1} such that their squares are equivalent (mod 2^n), n ≥ 4. Their sum will be equal to 2^{n-1} and so we can sieve over these values of x in the block, for 2^n, with regular intervals of 2^{n-1}.

Suppose that kN is even, then any even square is congruent to 0 or 4 (mod 8). So, if kN ≡ 2 (mod 4), then only 2 divides all even values of x for $x^2 - kN$. If kN ≡ 0 (mod 4), then 4 divides all even values of x for $x^2 - kN$. Sieving by higher powers of 2 can be done by using the fact that the period with which we have to sieve by 2^n, n ≥ 3 is given by $2^{[(n+1)/2]}$, where [(n+1)/2] denotes the greatest integer ≤ (n+1)/2. We will find all the roots of kN, that are below $2^{[(n+1)/2]}$ and then sieve over these values of x in the block for 2^n, with regular intervals of 2^{n-1}. This technique can be used to sieve by the powers of 2, and for all the powers of other primes, we can find the square root of the prime and then lift it by using the Hensel's Lemma for the prime powers.

In this version of quadratic sieve, we use the single polynomial $x^2 - kN$ (mod p). The problem with using this single polynomial is that the values of $x^2 - kN$ grow rapidly, as we move away from $x = \sqrt{kN}$. Actually, we assume that we are using the polynomial $(x+b)^2 - kN$, starting with $b = [\sqrt{kN}]+1$, for positive and negative values of x. As x grows larger, $(x+b)^2 - kN \approx x^2 + 2x\sqrt{kN}$. As x grows larger, it is less likely for the relations to be smooth, and therefore it is more difficult to collect any relations.

3.4.1 Multiple Polynomial Quadratic Sieve

The problem with $(x+b)^2 - kN$ less likely to be smooth, as x grows larger, can be overcome by using multiple polynomials, by keeping the relations as small as possible. Let us for simplicity purposes, keep k = 1, and we sieve polynomials of the form $(ax+b)^2 - N = a^2x^2 + 2axb + b^2 - N$, such that $0 < b < a$. We sieve this polynomial for x in the closed interval [-M, M]. Then the largest values of $(ax+b)^2 - N$ are given by $a^2M^2 - N$ and the smallest values will be –N. We choose $a \approx (\sqrt{2N})/M$ to make sure that the largest and smallest values are nearly of the same size in magnitude.

We choose b such that $b^2 - N$ is divisible by a, say $b^2 - N = ac$, then $(ax+b)^2 - N = a^2x^2 + 2axb + ac$, so we have that $(ax+b)^2 - N = a(ax^2+bx+c)$. Thus, every value of $(ax+b)^2 - N$ will be divisible by a. Suppose that $a = q^2$, for some integer q. Then, we have that $(ax+b)^2 \equiv q^2(ax^2+bx+c) \bmod N$. So, $(ax+b)^2 \bmod N$ is smooth if and only if (ax^2+bx+c) is smooth. We have that, $[(ax+b)q^{-1}]^2 \equiv (ax^2+bx+c) \bmod N$. We sieve this polynomial for all the factor base primes p, such that N is a quadratic residue (mod p), starting with some prime $q \approx \sqrt{(\sqrt{(2N)}/M)}$ such that N is a quadratic residue (mod q). Then, if a prime p divides $(ax+b)^2 - N$, we have $x = a^{-1}(\pm\sqrt{N} - b) \bmod p$, and we can use the above Hensel's Lemma to lift the roots of the prime powers. We can add the logarithm of the prime or prime power, starting from 0, to the sieve location, if p^n divides $(ax+b)^2 - N$, and since $(ax+b)^2 - N = q^2(ax^2+bx+c)$, so that to flag the smooth relations, we should check for those sieve locations that have accumulated a value of atleast $\log(q^2(ax^2+bx+c))$.

The following table gives an optimal value of factor base size and value of M needed for numbers upto 66 digits in size. [10]

Digits	Factor Base Size	M
24	100	5000
30	200	25000
36	400	25000
42	900	50000
48	1200	100000
54	2000	250000

60	3000	350000
66	4500	500000

Table 3.1: Optimal Factor Base Size and Block Size for MPQS

3.4.2 Self Initiating Quadratic Sieve

The self initiating quadratic sieve provides a fast way to change polynomials. This uses a smaller M value than multiple polynomial quadratic sieve, and since M values are smaller, the residues are smaller and are more likely to be smooth. This is why SIQS can be expected to be more significantly faster than MPQS. [12]

SIQS uses the same polynomials as MPQS, namely $(ax+b)^2 - N$, with $b^2 - N = ac$. So, again we have that $(ax+b)^2 - N = a(ax^2+bx+c)$. In MPQS, we took a to be a square of a large prime. In SIQS, we take a to be the product of several factor base primes. So, again we have that $(ax+b)^2 - N$ is smooth if and only if (ax^2+bx+c) is smooth. This allows us to get several polynomials for each value of a, since there can be several choices of b for every choice of a. Suppose that $a = q_1q_2q_3\ldots q_s$, where the q_i's are distinct primes in the factor base. To generate a polynomial, we have to find out a value of b, such that $b^2 \equiv N \pmod{a}$. There are actually 2^s distinct values of b that satisfy this condition, but only half of the values of b will be used since $(ax+b)^2 - N$ gives the same residues as $(ax-b)^2 - N$. So, we can use 2^{s-1} distinct polynomials for every choice of a.

We need to construct integers B_i, $1 \leq i \leq s$, such that $B_i^2 \equiv N \pmod{q_i}$, and that $B_i \equiv 0 \pmod{q_j}$, for $1 \leq j \leq s$, $i \neq j$. By the Chinese Remainder Theorem, the square of any combination of $\pm B_1 \pm B_2 \pm B_3 \ldots \pm B_s$ will be congruent to N (mod a). Since, we want only one value of b or –b pair, we can fix the sign of B_s to be positive. Now, that the values of B_i are given by using the formula $(a/q_i)\gamma_{a,q_i}$, where γ_{a,q_i} is the smallest positive residue of $t_{q_i}(a/q_i)^{-1} \bmod q_i$ and $t_{q_i}^2 \equiv N \pmod{q_i}$. Actually, there are two such choices for t_{q_i}. We will choose the one that makes the value of γ_{a,q_i} to be smaller.

We take the value of b_1 to be equal to $B_1+B_2+B_3+\ldots+B_s \pmod{a}$. The remaining values of b can be calculated quickly by applying a formula for Gray Code.

$b_{i+1} = b_i + 2(-1)^{\lceil i/2^v \rceil} B_v$, where 2^v divides 2i for i = 1 to $2^{s-1} - 1$.

CHAPTER 4
PRIMALITY TESTS

4.1 FERMAT'S LITTLE THEOREM

The most basic method to test whether a number is prime is the Fermat's Little Theorem. If p is prime, then $a^{p-1} = 1 \pmod{p}$. We will assume throughout that a and p are relatively prime numbers. The converse of this theorem can be used as a primality test. If $a^{p-1} = 1 \pmod{p}$, then p may be prime. The fact that p is prime, will not always be the case. Though if $a^{p-1} \neq 1 \pmod{p}$, then p is certainly composite, but if $a^{p-1} = 1 \pmod{p}$, then p is not always prime. Composite numbers p, such that $a^{p-1} = 1 \pmod{p}$, are called pseudo primes base a. If p satisfies $a^{p-1} = 1 \pmod{p}$, then we will call p as a probable prime (PRP) base a.

For every base a, there will be infinitely many pseudo primes, each base containing a different set of values. For example, 341 is a PRP base 2, 91 is a PRP base 3, 217 is a PRP base 5, and 25 is a PRP base 7. So, to increase our suspicion that the number is prime, we can test for more than one base. It would be nice if it would confirm the number is prime.

Unfortunately, there exist numbers that are pseudo primes for every base. Such numbers are called Carmichael Numbers. Recently it has been proved that there are infinitely many Carmichael Numbers. Below 10^{15}, there are 105212 Carmichael Numbers, and the density increases as we go higher and higher.

But, there are Carmichael numbers with no small factors. A Carmichael number can be recognized easily by its prime factorization. An integer $N > 1$ is a Carmichael number if and only if it is composite, square free and for every prime p dividing N, we have p-1 dividing N-1. The Carmichael Numbers below 100000 are given by 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 46657, 52633, 62745, 63973, 75361.

4.2 RABIN MILLER'S TEST

A better idea to check whether a number is prime, besides checking more than one base, is to check the values of $a^{(p-1)/2}$, $a^{(p-1)/4}$, $a^{(p-1)/8}$, etc. Note that if $a^{(p-1)/2^r} = 1 \pmod p$, then since p is prime, $a^{(p-1)/2^{r+1}} = \pm 1 \pmod p$. Otherwise, p is composite. The advantage of this test is that fewer composite numbers pass this test.

For example, $2^{170} = 1 \pmod{341}$ and $2^{85} = 32 \pmod{341}$. So, 341 is not prime. Indeed, we can derive its factorization from these equations. Note that $32^2 = 1 \pmod{341}$, so factors are GCD(32-1, 341) = 31 and GCD(32+1, 341) = 11. Alternately, $2^{280} = 1 \pmod{561}$ and $2^{140} = 67 \pmod{561}$. So, $67^2 = 1 \pmod{561}$. Thus, the factors are GCD(67-1, 561) = 33 and GCD(67+1, 561) = 17.

So, we call a number p to be a strong probable prime (SPRP) base a (assuming that a, p are relatively prime), if

(1) $a^{p-1} = 1 \pmod p$

(2) Write $p-1 = 2^s t$, where t is odd. p is SPRP base a, if either $a^t = 1 \pmod p$ or $a^{2^i t} = -1 \pmod p$, for some i with $0 \le i \le s-1$.

But, this test will not prove that the number is prime. For example, 2047 is a SPRP base 2, 121 is a SPRP base 3, 781 is a SPRP base 5, and 91 is a SPRP base 10. Every Mersenne Number and every Fermat Number is a SPRP base 2. Although, using this test for more than one base will give us a fairly good idea that the number is prime. But, that is not enough. We have a few results.

If $n < 1{,}373{,}653$ is a both 2 and 3-SPRP, then *n* is prime

If $n < 25{,}326{,}001$ is a 2, 3 and 5-SPRP, then *n* is prime

If $n < 118{,}670{,}087{,}467$ is a 2, 3, 5 and 7-SPRP, then either $n = 3{,}215{,}031{,}751$ or *n* is prime

If $n < 2{,}152{,}302{,}898{,}747$ is a 2, 3, 5, 7 and 11-SPRP, then *n* is prime

If $n < 3{,}474{,}749{,}660{,}383$ is a 2, 3, 5, 7, 11 and 13-SPRP, then *n* is prime

If $n < 341{,}550{,}071{,}728{,}321$ is a 2, 3, 5, 7, 11, 13 and 17-SPRP, then *n* is prime

If $n < 9{,}080{,}191$ is a both 31 and 73-SPRP, then n is prime.
If $n < 4{,}759{,}123{,}141$ is a 2, 7 and 61-SPRP, then n is prime.

If $n < 170{,}584{,}961$ is a 350 and 3958281543-SPRP, then n is prime.
If $n < 75{,}792{,}980{,}677$ is a 2, 379215, and 457083754-SPRP, then n is prime.
If $n < 21{,}652{,}684{,}502{,}221$ is a 2, 1215, 34862, and 574237825-SPRP, then n is prime.

The following number was given by Arnault in his Ph.D. thesis is a SPRP to 46 bases, and is not prime.
80383745745363949125707961434194210813883768828755814583748891752229742737653336521865023361639600454579150420236032087665699667609872840439654082329287387918508691668573282677617710293896977394701670823042868710999743997654414484534115587245063340927902227529622941498423068816854043264575340183297861112989606448452161916528725975349 01

CHAPTER 5
PRIMALITY PROVING

The algorithms stated in the previous chapters, give an idea about whether the number is prime, but does not prove it. In order to prove that the number is prime, we need special algorithms, but for special numbers, polynomial time algorithms do exist.

We will start with the test by Lucas.
If $a^{p-1} = 1 \pmod{p}$, but $a^{(p-1)/q} \neq 1 \pmod{p}$, for every prime q dividing p-1, then p is prime. That means that the order of a (mod p) is not a proper divisor of p-1, but is equal to p-1. Since the order of a for any number (mod p) is φ(p), this states that φ(p) cannot be less than p-1. So, φ(p) = p-1 and that p is prime.

The problem with this to be used as a primality test is not because of finding some value of a satisfying this, but the difficulty in obtaining the complete factorization of p-1. As the number goes bigger, the lesser is the chance that p-1 will be smooth, and so finding out the penultimate prime factor of p-1 will be very difficult. However, if the factorization of p-1 is smooth, such as for Fermat Numbers in which p-1 is always a power of 2, then this test can be used as a primality test. Note that for Fermat Numbers, the base a can be chosen such that a is a quadratic non-residue (mod F_n), such as a = 3. Note that only the prime 2 divides $F_n - 1$.

5.1 PÉPIN'S TEST (FOR FERMAT NUMBERS)

$F_n = 2^{2^n} + 1$ is prime if and only if $3^{(F_n-1)/2} = -1 \pmod{F_n}$

The largest number tested to be composite by using this algorithm in 2000 is F_{24}. The above test can be extended so that we can prove the primality of a number, even only if a partial factorization of p-1 is known.

Pocklington Theorem: If only a partial factorization of p-1 is known, such that p-1 = F.R, where F is completely factored, and R is un-factored composite. Then if $a^{p-1} = 1 \pmod{p}$

and gcd($a^{(p-1)/q} - 1$, p) = 1 for every prime q dividing F, then every prime factor of p is congruent to 1 (mod F). Further, if $F > \sqrt{p}$, then p is prime.

5.2 PROTH'S THEOREM

For numbers of the form $k.2^n + 1$, where factorization of p-1 is smooth, for small values of k, the above theorem can be used. If $p = k.2^n + 1$ such that $k < 2^n$, k is odd and $a^{(p-1)/2} = -1 \pmod{p}$ for some base a, then p is prime.

This theorem can be extended further such that if factorization of p-1 only upto $\sqrt[3]{p}$ is known, then it can be verified if p is prime. Let p-1 = F.R, where F is completely factored, and R is unfactored cofactor such that $\sqrt{p} > F \geq \sqrt[3]{p}$. Let c_1 and c_2 be integers such that $p = c_2F^2 + c_1F + 1$, where $0 \leq c_1, c_2 \leq F-1$. Then p is prime if and only if $c_1^2 - 4c_2$ is not a perfect square.

5.3 LUCAS LEHMER TEST (FOR MERSENNE PRIMES)

For Mersenne Numbers of the form $p = 2^n - 1$, one greater than it will immediately give a smooth factorization, that is a power of 2. For Mersenne Numbers, for which p+1 is smooth, we make use of Lucas Sequences to test if p is prime. Note that only prime 2 divides p+1 for Mersenne Numbers.

We recall that if p is prime, then $U_{p-(\Delta/p)} = 0 \pmod{p}$, where $\Delta = P^2 - 4Q$, the recurrence equation being $x^2 - Px + Q = 0$ and $U_n = (\alpha^n - \beta^n)/(\alpha - \beta)$ and $V_n = (\alpha^n + \beta^n)$, where α and β are the roots of the above equation. Similar to the Pocklington test, we can also have a similar test. Let $(\Delta/p) = -1$ and GCD(p, 2Q) = 1. If F is a divisor of p+1 and $U_{p+1} \equiv 0 \pmod{p}$, and $GCD(U_{(p+1)/q}, p) = 1 \pmod{p}$ for every prime q dividing F, then every prime m dividing p satisfies $m \equiv (\Delta/m) \pmod{F}$. Further, if $F > \sqrt{p} + 1$, then p is prime.

For Mersenne Numbers, we can use the equation $x^2 - 4x + 1 = 0$, and $p = 2^n - 1$ is prime if and only if $V_{2^{n-2}} = 0 \pmod{p}$. So, p is prime if and only if $(2+\sqrt{3})^{2^{n-2}} + (2-\sqrt{3})^{2^{n-2}} = 0 \pmod{p}$. Since $V_1 = 4$ and $V_{2n} = V_n^2 - 2$, we can use the following algorithm to test whether $p = 2^n - 1$ is prime.

Lucas Lehmer Test: Let $p = 2^n - 1$, and $S_1 = 4$.

For i = 2 to n-1, define $S_i = S_{i-1}^2 - 2 \pmod{2^n - 1}$.

$p = 2^n - 1$ is prime if and only if $S_{n-1} = 0$.

5.4 LUCAS LEHMER RIESEL TEST

The Lucas Lehmer Test can be extended further to prove that numbers of the form $p = h.2^n - 1$ are prime, where $h < 2^n$. It depends upon choosing a clever value for S_1. [21]

If h = 1, then $S_1 = 4$ will hold for all odd values of n, and $S_1 = 3$ will hold for all $n \equiv 3 \pmod 4$

If h = 3, then $S_1 = 5778$ can be used for all $n \equiv 0, 3 \pmod 4$

If $h = 6k \pm 1$, k is an integer, then $S_1 = (2 + \sqrt{3})^h + (2 - \sqrt{3})^h$ can be used for all the values of n. Note that, here we use the value of P = 4 and Q = 1 and calculate the value of V_h.

Otherwise, we are left with the case where h is a multiple of 3, and we have to be more careful in choosing the value of S_1.

Try with v_1 values starting from 3, and let D be the square free part of $v_1^2 - 4$.

Let $v_1 = \alpha + \alpha^{-1}$, and so v_1 satisfies the equation $\alpha^2 - v_1\alpha + 1 = 0$.

Now, that the values of a and r are obtained from the following equations:

$\alpha = (a + b\sqrt{D})^2/r$ and $r = |a^2 - b^2D|$, b = 1.

Let $k = \sqrt{[(v_1^2 - 4)/D]}$. Then, $\alpha = [v_1 + \sqrt{(v_1^2 - 4)}]/2 = [v_1 + k\sqrt{D}]/2$.

This corresponds to $[(a^2 + b^2D) + 2ab\sqrt{D}]/r$. So, that now we can now easily solve for a and r. These should satisfy the following equations:

$(D/N) = -1$, $(r/N)\ (a^2 - b^2D)/r = -1$. If not choose a different value for v_1.

Then, the starting value S_1 is taken to be $\alpha^h + \alpha^{-h}$, that is V_h with $P = v_1$ and Q = 1, satisfying the Lucas equation $x^2 - Px + Q = 0$. The value of V_h can be calculated by using the following recurrence equations.

$V_0 = 2$, $V_1 = P$, $V_n = PV_{n-1} - QV_{n-2}$ for all $n \geq 2$. $V_{2n} = V_n^2 - 2Q^n$, $V_{2n-1} = V_nV_{n-1} - PQ^{n-1}$.

5.5 DISTRIBUTED COMPUTING PROJECTS

There are two distributed computing projects, one each, which make use of Proth's Theorem and Lucas Lehmer Riesel Test for proving that numbers of the form $k.2^n + 1$ and $k.2^n - 1$ are prime respectively.

In 1960, Sierpinski proved that there exist infinitely many values of k for which $k.2^n + 1$ is composite for all n. Such values of k are called Sierpinski Numbers. In 1962, John Selfridge showed that k = 78557 is one such example. Seventeen or Bust is the distributed computing project that tries to prove that 78557 is the smallest Sierpinski Number, by finding out a value of n, for every value of k below 78557, such that $k.2^n + 1$ is prime. It is named so, because when the project started, there were 17 numbers below 78557, that were left to be examined. If any number exists below 78557 satisfying this condition, then it will be the smallest Sierpinski Number.

Similarly, in 1956, Riesel proved that there are infinitely many values of k for which $k.2^n - 1$ is composite for all n. These values of k are called Riesel Numbers. He showed that k = 509203 satisfy this condition. Riesel Sieve is the distributed computing project that tries to prove that 509203 is the smallest Riesel Number, by finding out a value of n, for every value of k below 509203, such that $k.2^n - 1$ is prime. If any number exists below 509203 satisfying this condition, then it will be the smallest Riesel Number.

CHAPTER 6
DISCRETE LOGARITHMS

6.1 PROBLEM STATEMENT

Given values of g, t, p, find out a value of l such that $g^l = t \pmod p$. Actually, since $g^{\varphi(p)} = 1 \pmod p$, so for any integral value of k, we will have $g^{l+k\varphi(p)} = t \pmod p$. So, there will be infinitely many values of l, satisfying this condition, if there exist one, and the problem will be to find out the lowest value of such l. The period is the gap after which the values of t repeat, and the period will be a divisor of φ(p). The largest value of period will be φ(p), and the value of g, which has the period of φ(p) is called the generator of the group (mod p), and the group is cyclic if p is prime and φ(p) = p-1. For a generator (or primitive root) (mod p), every value of t such that $1 \leq t \leq p-1$ has a solution for l, such that $1 \leq l \leq p-1$.

6.1.1 Calculating the period

Calculating the period is quite obvious. First of all, you will need to factor φ(p). Initially, assuming that the period is φ(p), then for every prime factor of φ(p), you will need to divide that period by the prime factor. Value = period / prime factor. If you get $g^{Value} = 1 \pmod p$, or $g^{2*Value} = g^{Value} \pmod p$, if p is not prime, then the new period = old period / prime factor. Repeat this process for all the prime factors of φ(p).

6.2 ALGORITHMS FOR SOLVING THE DISCRETE LOGARITHM PROBLEM

Assuming that on an initial attempt, any person uses brute force techniques for small values of p, to search for a value of l, such that $g^l = t \pmod p$, starting with l = 1, we will start to look at the better algorithms to solve this problem, starting up with the Monte Carlo Methods.

6.2.1. POLLARD'S RHO ALGORITHM

The Pollard's Rho approach can also be used to solve the Discrete Logarithm problem. [19] Let $x_i = t^{a_i}g^{b_i} \pmod p$ and $a_0 = 0$, $b_0 = 0$, $x_0 = 1$. Here, we define a pseudo random function such as:

$(a_{i+1}, b_{i+1}) = (a_i + 1 \pmod{\varphi(p)}, b_i)$, if $0 < x_i < p/3$

$(a_{i+1}, b_{i+1}) = (2a_i \pmod{\varphi(p)}, 2b_i \pmod{\varphi(p)})$, if $p/3 < x_i < 2p/3$

$(a_{i+1}, b_{i+1}) = (a_i, b_i + 1 \pmod{\varphi(p)})$, if $2p/3 < x_i < p$

And so, we have that

$x_{i+1} = tx_i \pmod{p}$, if $0 < x_i < p/3$

$x_{i+1} = x_i^2 \pmod{p}$, if $p/3 < x_i < 2p/3$

$x_{i+1} = gx_i \pmod{p}$, if $2p/3 < x_i < p$

We compare the different values of x_i to find out a match. Similar to the tortoise-hare algorithm (Floyd's cycle finding algorithm), we compare x_1 with x_2, x_2 with x_4, x_3 with x_6 and so on. We compare x_m with x_{2m} each time, to look for a match. We have two variables for x, a and b, each of which hold the iterations for m and 2m respectively.

If a match is found out, say $x_j = x_k$, then we have $t^{a_j}g^{b_j} = t^{a_k}g^{b_k} \pmod{p}$. Since the period is $\varphi(p)$, or a divisor of $\varphi(p)$, comparing the powers of t and g, and writing $t = g^l$, we will have $(a_j - a_k).l \equiv b_k - b_j \pmod{\varphi(p)}$, from which we could calculate the value of l as to be $l = (b_k - b_j)(a_j - a_k)^{-1} \pmod{\varphi(p)}$.

6.2.2. POLLARD'S LAMBDA ALGORITHM

Pollard's Lambda algorithm is a distributed computing algorithm, in which multiple computers (or processes) participate over a network. Each node starts with some value t^r for some random value of r, and uses the same pseudo random function, $f: G \rightarrow S$, where S is a relatively small set of integers, whose mean value is comparable to the size of group G. The powers g^s for the values of s in set S are pre-computed. Then, the walk starting from t^r is, $w_0 = t^r$, $w_1 = w_0 g^{f(w_0)}$, $w_2 = w_1 g^{f(w_1)}$ and so on. If another node starting at r', and walking through the sequence w_0', w_1', w_2'… has a collision with the sequence w_0, w_1, w_2, etc., that is $w_i' = w_j$ for some i, j, then

$t^{r'} g^{f(w_0')+f(w_1')+\ldots+f(w_{i-1}')} = t^r g^{f(w_0)+f(w_1)+\ldots+f(w_{j-1})}$

So, since $t = g^l$, we can substitute it and calculate l as follows:

$(r - r')l = [f(w_0) + f(w_1) + \ldots + f(w_{j-1})] - [f(w_0') + f(w_1') + \ldots + f(w_{i-1}')] \pmod{\varphi(p)}$

6.2.3 BABY STEPS, GIANT STEPS

We will look at the third algorithm for solving the discrete logarithm problem. We have to find out a value for l, such that $g^l = t \pmod p$, and whatever be the value of l, and if it satisfies this condition, then the lowest possible value of l can be given by l (mod period).

Let b = Ceiling($\sqrt{p}$), and we will write l in base $b = l_0 + l_1 b$, where $0 \leq l_0, l_1 \leq b-1$. We have $g^{l_1 b} = tg^{-l_0} = th^{l_0}$, where $h = g^{-1} \pmod p$. So, we will take two lists, one containing the values of (g^{ib}) for $0 \leq i \leq b-1$, and the other list containing the values of (th^j) for $0 \leq j \leq b-1$. We have to search for a collision, and the technique is to use some sorting algorithm, and traverse the sorted lists to look for a collision. Sorting can be done in worst case in O(b ln b) time by using quick sort or merge sort, and the computations of values and collision searches can be done in O(b) time, and so the overall time complexity of the algorithm will be O(b) = O($\sqrt{p}$). All the above three algorithms have running time of O($\sqrt{p}$), and so it is exponential.

Once the collision has been detected, we have $g^{ib} = th^j$, so writing $t = g^l$ and $h = g^{-1}$, we have l = j + ib.

6.2.4 POHLIG HELLMAN ALGORITHM

This algorithm has better running time over the above three algorithms, although it is exponential only. The algorithm is especially fast, if the factorization of φ(p) is smooth, and if the largest prime factor of φ(p) is, say, M, then the algorithm requires O($\sqrt{M}$) operations. [30] Since with the modern computers, around 10^8 operations are feasible in one minute, the algorithm completes in a reasonable amount of time if the largest prime factor of φ(p) lies below 10^{16}.

Let $\varphi(p) = q_1^{a_1}.q_2^{a_2}\ldots q_n^{a_n}$, where $q_1, q_2, \ldots, q_n$ are prime numbers. We have to calculate the value of x such that $g^x = t \pmod p$. The technique is to find out the value of $x_i \pmod{q_i^{a_i}}$, for i = 1, 2, …, n, and then make use of the Chinese Remainder Theorem to find out the value of x. To find out the value of $x_i \pmod{q_i^{a_i}}$, we use the following techniques:

If $a_i = 1$, and so the power of prime is 1, we calculate $x_i \pmod{q_i}$

We write $x_i = c_0 + c_1 q_i$, where $0 \leq c_0 < q_i$.

We get that, $t^{\varphi(p)/q_i} = (g^x)^{\varphi(p)/q_i} = (g^{c_0 + c_1 q_i})^{\varphi(p)/q_i} = (g^{\varphi(p)/q_i})^{c_0} . g^{c_1 \varphi(p)} = (g^{\varphi(p)/q_i})^{c_0} \pmod{p}$, since $g^{k\varphi(p)} \equiv 1 \pmod{p}$. We can now calculate the value of c_0 such that $t^{\varphi(p)/q_i} = (g^{\varphi(p)/q_i})^{c_0} \pmod{p}$, in the smaller field of q_i elements, by using Pollard's Rho algorithm or Baby Steps, Giant Steps. Or we can calculate the values of $(g^{\varphi(p)/q_i})^k$ for $0 \leq k < q_i$, if q_i is smaller, and find out the value of c_0. We have that, $x_i \equiv c_0 \pmod{q_i}$

If $a_i = 2$, the power is 2, and we need to calculate the value of $x_i \pmod{q_i^2}$

Write $x_i = c_0 + c_1 q_i + c_2 q_i^2$, where $0 \leq c_0, c_1 < q_i$. We can find out c_0 by using the above method. We have that, $t^{\varphi(p)/q_i^2} = (g^x)^{\varphi(p)/q_i^2} = (g^{c_0 + c_1 q_i + c_2 q_i^2})^{\varphi(p)/q_i^2} = (g^{\varphi(p)/q_i^2})^{c_0} . (g^{\varphi(p)/q_i})^{c_1} \pmod{p}$. Substituting the value of c_0, we can calculate the value of c_1. We observe that $x_i \equiv c_0 + c_1 q_i \pmod{q_i^2}$.

If $a_i = m > 2$, then similarly calculate the value of $x_i \pmod{q_i^m}$.

Write $x_i = c_0 + c_1 q_i + c_2 q_i^2 + \ldots + c_m q_i^m$. We need to find out $c_0, c_1, c_2, \ldots, c_{m-2}$ in order. Then, we have $t^{\varphi(p)/q_i^m} = (g^{\varphi(p)/q_i^m})^{c_0} . (g^{\varphi(p)/q_i^{m-1}})^{c_1} . \ldots . (g^{\varphi(p)/q_i^2})^{c_{m-2}} . (g^{\varphi(p)/q_i})^{c_{m-1}} \pmod{p}$. Substituting those already calculated values of $c_0, c_1, c_2, \ldots, c_{m-2}$, we calculate the value of c_{m-1}. We infer that, $x_i \equiv c_0 + c_1 q_i + c_2 q_i^2 + \ldots + c_{m-1} q_i^{m-1} \pmod{q_i^m}$.

Once we have the various values of $x_i \pmod{q_i^{a_i}}$ for i = 1 to n, we can make use of the Chinese Remainder Theorem to calculate the value of x. Let $r_i = q_i^{a_i}$, where we have the various r_i's for $1 \leq i \leq n$, to be relatively prime. We have the different congruences, for $x \equiv x_i \pmod{r_i}$. We have $p = r_1 r_2 r_3 \ldots r_n$. For i = 1 to n, we calculate $p_i = p/r_i$ and $v_i = p_i^{-1} \pmod{r_i}$. Then, from these values, we calculate $x = x_1 p_1 v_1 + x_2 p_2 v_2 + \ldots x_n p_n v_n \pmod{p}$.

6.2.5 INDEX CALCULUS METHOD

The index calculus method for calculating discrete logarithms has sub-exponential running time when compared to the exponential running time against the other algorithms. The basic idea is similar to the quadratic sieve for integer factorization, where you choose a factor base composed of small primes, and filter those values of $g^x \pmod p$, which are smooth within the factor base primes. By elementary linear algebra, if the factor base consists of M primes, you will need atleast M+1 relations at the end, to guarantee a linear dependency. Then, you will find a value of r, for which t^r is smooth within the factor base primes, and try to express it as a linear combination of various values of g^x. Thus, if $t^r \equiv g^{x_1+x_2+x_3+\ldots+x_N} \pmod p$, we can calculate the value of $t^r = (g^x)^r = g^{x_1+x_2+x_3+\ldots+x_N} \pmod p$. Hence, we have that, $x = (x_1+x_2+x_3+\ldots+x_N).r^{-1} \pmod{\varphi(p)}$.

Unlike the quadratic sieve, the difficulty with this algorithm is with the matrix. In the quadratic sieve, we have all matrix entries to be composed of only bits containing 0 or 1, but in the index calculus method, we have all the entries to be integers, whose size is comparable to the value of $\varphi(p)$. Clearly, we solve the matrix $(\text{mod } \varphi(p))$, or atleast (mod Period), since the value of the period can be easily calculated earlier. So, that we have to apply Gaussian Elimination carefully while solving the matrix and it takes extra time and effort in solving the matrix.

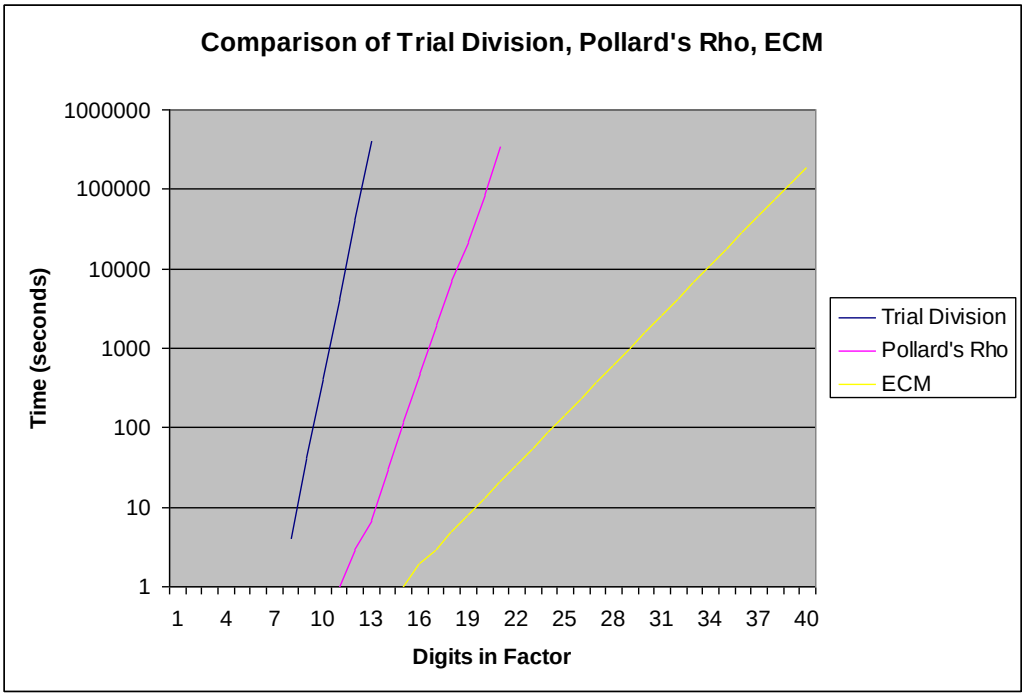
CHAPTER 7

SUMMARY AND CONCLUSION

The algorithms Trial Division, Pollard's Rho, ECM were implemented and the time taken for factoring numbers, with factors having upto 40 digits was studied and the results tabulated.

Digits in Factor	Trial Division (sec)	Pollard's Rho (sec)	ECM (sec)
10	420	1	
13	420000	7	
15	42000000	120	1
17		3600	3
20		43200	10
25			154
30			1163
35			10296
40			72108

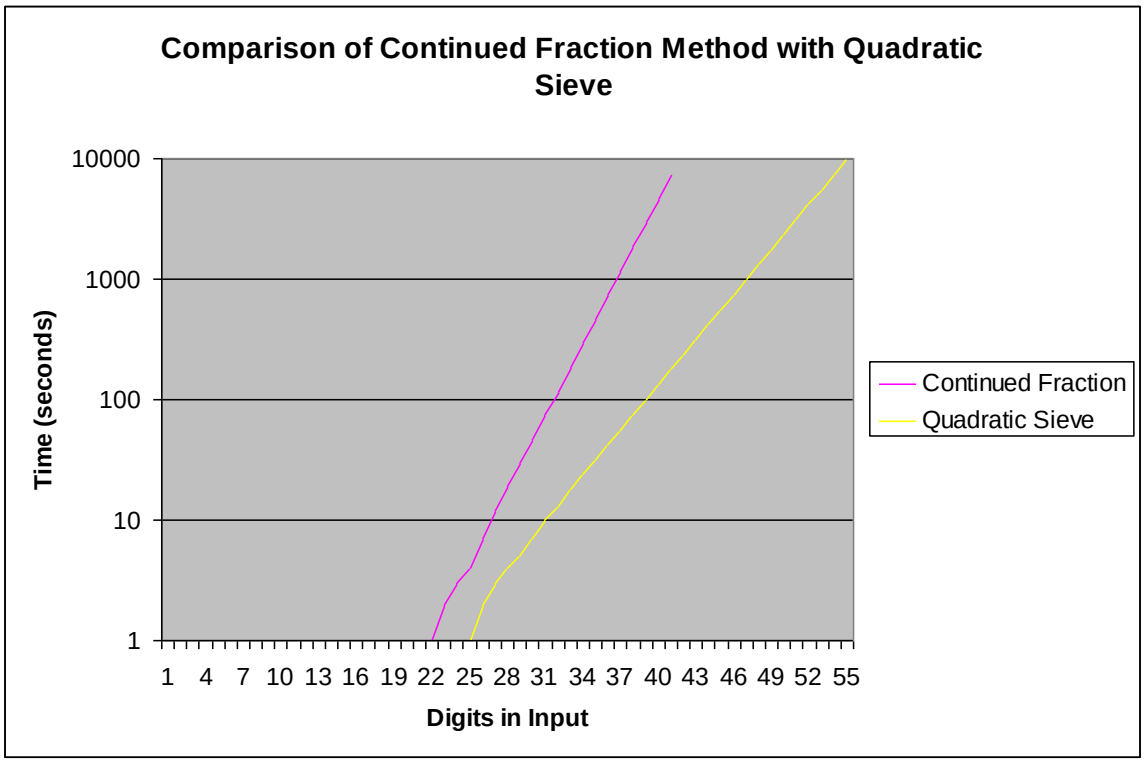
Table 7.1 Comparison of time taken for Trial Division, Pollard's Rho, ECM



The Continued Fraction and Quadratic Sieve Methods were implemented and the time taken for factoring numbers, with upto 55 digits was studied and the results tabulated.

Digits in Input	Continued Fraction Method (sec)	Quadratic Sieve (sec)
25	7	1
30	76	10
35	600	30
40	3000	95
45	21600	360
50		1800
55		10800

Table 7.2 Comparison of time taken for CFRAC with Quadratic Sieve



7.1 SCOPE FOR FUTURE WORK

The Quadratic Sieve is not the best known factoring algorithm for general purpose numbers with two large prime factors, but the Number Field Sieve has better asymptotic running time. In fact, the cross over point between the Quadratic Sieve and Number Field

Sieve is about 97 digits, above which the Number Field Sieve is faster than the Quadratic Sieve. To be precise, Continued Fraction Method has running time of $e^{\sqrt{(2 \log n \log \log n)}}$, the Quadratic Sieve has running time of $e^{\sqrt{(\log n \log \log n)}}$ and the Number Field Sieve has running time of $e^{c(\log n)^{1/3}(\log \log n)^{2/3}}$. The Number Field Sieve uses a higher degree polynomial, and it comes in two variations, the Special Number Field Sieve and General Number Field Sieve. The Special Number Field Sieve can only be used for numbers of special algebraic form, such as Cunningham Numbers [31], which are of the form $b^n \pm 1$, and in general, it should be of the form $q_0 + q_1x + q_2x^2 + \dots q_nx^n$, for small values of q_i, if a degree n polynomial is used. For SNFS, degree 4 polynomial is used for 100 digits, degree 5 for 150 digits and degree 6 is optimal for 200 digits. For example, in 1990, the ninth Fermat number $F_9 = 2^{512}+1$ was completely factored by using the Special Number Field Sieve. [13] If a general number can be expressed in the above form, with larger values of q_i, then it becomes the General Number Field Sieve, differing only in the polynomial selection method, with the larger values of coefficients q_i, and this algorithm can be used for any general number. [17] In fact SNFS is about 1.43 times faster than GNFS for a number of the same size.

Further, Shor's algorithm is a polynomial time factoring algorithm on a quantum computer. It is based upon finding out the period p for N, by using quantum computations, such that $a^p \equiv 1 \pmod N$, where a is chosen such that it is relatively prime to N. Then, we can calculate the value of $\varphi(N)$, which will be a multiple of the period, and after that we can easily extract the factors of N. [15]

For discrete logarithms, Index Calculus Method has sub-exponential running time. For primality proving, we can work on APRT-CLE [20] and Elliptic Curve Primality Proving. Elliptic Curve Primality Proving is based upon the fact that if s divides k, and that $[k]P = O \pmod p$, but $[k/q]P \neq O \pmod p$ for every prime q dividing s, then every prime m dividing p satisfies, $[\#E]F_m \equiv 0 \pmod s$. Further, if $s > \sqrt{p}$, then p is prime. Actually, the techniques to find out the group order of the elliptic curve digs deep into polynomial factorization, so that we will work upon it later on.

In August 2002, two IIT Kanpur students: Kayal and Saxena, and their professor: Agrawal, introduced a new deterministic polynomial time algorithm for proving primality, that does not lie on any unproved assumptions, such as the Riemann Hypothesis. [14] It states that p is prime if and only if $(x-a)^p = x^p - a \pmod p$. Here x and a are not simple constants, but it should be treated as a polynomial variable, which is a bit confusing, so we will analyze about it and work upon it later on.

In addition, for the already implemented algorithms, for the Linear Algebra of Quadratic Sieve and other allied algorithms, there are better algorithms than Gaussian Elimination for solving sparse matrices over GF(2). One of them is Block Lanczos, [16] which can be implemented on a single computer, and Block Weidemann, which is efficient as a distributed computing algorithm.

For algorithms such as p-1, p+1 and ECM, we have to multiply the initial point with every prime less than B1, and for every prime, we should use the largest power of prime less than B1. There are very efficient algorithms for Big Integer multiplication, such as Fast Fourier Transform algorithms. For Big Integer squaring, such as in Lucas Lehmer Test and Pépin's test, we can use Discrete Weighted Transform algorithms. For these two algorithms, it is very convenient to keep $2^n - 1$ and $2^{2^n} + 1$, as a string of bits, in base 2, consisting only of zeros and ones, and it is very comfortable to square or multiply $\pmod{2^n - 1}$ or $\pmod{2^{2^n} + 1}$. Keeping these numbers in base 10, and doing such calculations, in decimal would take a lot of time.

REFERENCES

Papers:

1. "A Tale of Two Sieves" by Carl Pomerance
2. "Factorization of the Eighth Fermat Number" by Richard Brent and John Pollard
3. "An Improved Monte Carlo Factorization Algorithm" by Richard Brent
4. "Factorization of the Tenth Fermat Number" by Richard Brent
5. "A p+1 method of Factoring" by Hugh Williams
6. "A method of Factoring and the Factorization of F7" by Michael Morrison and John Brillhart
7. "An FFT extension to the p-1 Factoring algorithm" by Peter Montgomery and Robert Silverman
8. "Speeding the Pollard and Elliptic Curve Methods of Factorization" by Peter Montgomery
9. "A practical analysis of the Elliptic Curve Factoring Algorithm" by Robert Silverman and Samuel Wagstaff
10. "The Multiple Polynomial Quadratic Sieve" by Robert Silverman
11. "How was F6 factored?" by Hugh Williams
12. "Factoring Integers with the Self Initializing Quadratic Sieve" by Scott Contini
13. "The Factorization of the Ninth Fermat Number" by Arjen Lenstra, Hendrik Lenstra, Mark Manasse and John Pollard
14. "Primes is in P" by Manindra Agrawal, Neeraj Kayal and Nitin Saxena
15. "Quantum Computing and Shor's algorithm" by Matthew Hayward
16. "A Block Lanczos Algorithm for finding dependencies over GF(2)" by Peter Montgomery
17. "On Polynomial Selection for the General Number Field Sieve" by Thorsten Kleinjung
18. "A kilobit Special Number Field Sieve Factorization" by Arjen Lenstra, Kazumaro Aoki, Jens Franke, Thorsten Kleinjung and Dag Arne Osvik
19. "Monte Carlo Methods for Index Computation (mod p)" by John Pollard
20. "On distinguishing prime numbers from composite numbers" by Leonard Adleman, Carl Pomerance and Robert Rumely
21. "Lucasian Criteria for the Primality of $N = h \cdot 2^n - 1$" by Hans Riesel
22. "Factoring Integers with Elliptic Curves" by Hendrik Lenstra

23. “The Number Field Sieve” by Steven Byrnes
24. “Modular Multiplication without Trial Division” by Peter Montgomery
25. “Aliquot Sequence 3630 ends after reaching 100 digits” by Manuel Benito, Wolfgang Creyaufmüller, Juan Varona and Paul Zimmermann
26. “A Beginner’s Guide to the General Number Field Sieve” by Michael Case
27. “The Multiple Polynomial Quadratic Sieve” by Brandt Kurowski
28. “Factoring Large Numbers with a Quadratic Sieve” by Joseph Gerver
29. “Theorems on Factorization and Primality Testing” by John Pollard
30. “An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance” by S.Pohlig and M.Hellman
31. “Factorizations of $b^n \pm 1$, b = 2, 3, 5, 6, 7, 10, 11, 12 up to high powers” by John Brillhart, Dick Lehmer, John Selfridge, Bryant Tuckerman and Samuel Wagstaff, Jr.
32. “Factorization of the Eleventh Fermat Number” by Richard Brent

Books:

1. “The Art of Computer Programming” by Donald Knuth, Volume 2: Semi-Numerical Algorithms
2. “Prime Numbers - A Computational Perspective” by Richard Crandall and Carl Pomerance
3. “The Development of The Number Field Sieve” by Arjen Lenstra, Hendrik Lenstra, Mark Manasse and John Pollard
4. “Prime Numbers and Computer Methods for Factorization” by Hans Riesel
5. “The Arithmetic of Elliptic Curves” by Joseph Silverman and John Tate

APPENDIX A

NUMBER THEORY CONCEPTS

A.1 GCD AND INVERSE

If x, y are two numbers, then gcd(x, y) represents the greatest common divisor of x and y. The gcd of two numbers can be calculated by the following code.Let x, y be two numbers such that $x \geq y$

```
while (y > 0)
{
(x, y) = (y, x mod y)
}
return x
```

Bezout's Theorem

If x, y are two numbers such that GCD (x, y) = g, then there exist integers a, b such that ax + by = g. The constants a, b can be calculated by the following algorithm.

```
(a, b, g, u, v, w) = (1, 0, x, 0, 1, y)
while (w > 0)
{
q = Integer Part of (g / w)
(a, b, g, u, v, w) = (u, v, w, a - qu, b - qv, g - qw)
}
return (a, b, g)
```

So, if you want to calculate inverse of x (mod y), that is, the number x^{-1} such that $x \times x^{-1} = 1 \pmod{y}$, then, it is a. And the inverse of y (mod x) is b.

The inverse can be calculated more easily by using the following formula.

$x^{-1} \pmod{y} = x^{\varphi(y) - 1} \pmod{y}$, where φ is the Euler's Phi Function.

A.2 CHINESE REMAINDER THEOREM

Consider the following set of congruences.

$x = n_1 \pmod{p_1}$

$x = n_2 \pmod{p_2}$

$x = n_3 \pmod{p_3}$

...

$x = n_r \pmod{p_r}$, where $p_1, p_2, p_3, \ldots p_r$ are pairwise relatively prime, i.e. $\gcd(p_i, p_j) = 1$ for all $1 \leq i < j \leq r$ and LCM $(p_1, p_2, p_3, \ldots p_r) = p_1p_2p_3\ldots p_r$

The Chinese Remainder Theorem claims that x has a unique solution $\pmod{p_1p_2p_3\ldots p_r}$ under these set of conditions such that $0 \leq x < p_1p_2p_3\ldots p_r$

For $1 \leq i \leq r$, if $m_i = p_1p_2p_3\ldots p_r / p_i$ and $v_i = m_i^{-1} \pmod{p_i}$, then x is given by

$x = n_1m_1v_1 + n_2m_2v_2 + n_3m_3v_3 + \ldots + n_rm_rv_r \pmod{p_1p_2p_3\ldots p_r}$

A.3 MODULAR SQUARE ROOT ALGORITHM

Here computing the square root of a number modulo a prime number is discussed.

Let a be the given number and p be the given prime number which forms the modulo for the calculations. If there exists a number x such that $x^2 = a \pmod{p}$, then x is called a square root of a (mod p)

If there exists a square root for a (mod p), then a is called quadratic residue (mod p), then there will be two square roots for a (mod p), such that their sum will be p.

If there exist no square root for a (mod p), then a is called quadratic non-residue (mod p)

In short, half of a less than p, have two square roots, and half of a less than p, have no square roots.

The Legendre Symbol (a/p) is defined by the following statements

1 if a is quadratic residue (mod p)

-1 if a is quadratic non-residue (mod p)

0 if a and p are not relatively prime.

For any prime number p and a base number a, the Legendre Symbol can be calculated by using the following formula

$(a/p) \equiv a^{(p-1)/2} \pmod p$

Note that, if p is prime number and a is less than p, by the Fermat's Little Theorem, $a^{p-1} \equiv 1 \pmod p$. Since p is prime, $a^{(p-1)/2}$ should be either 1 (mod p) or -1 (mod p).

If m is a composite number (or a prime number with only one prime factor), and has prime factorization $p_1^b \times p_2^c \times p_3^d \times ... \times p_n^z$ then the Jacobi Symbol (a/m) is defined by $(a/p_1)^b \times (a/p_2)^c \times (a/p_3)^d \times ... \times (a/p_n)^z$.

So, given a base number a and a prime number p, whether a has a square root (mod p), or not can be calculated by the Legendre Symbol $(a/p) = a^{(p-1)/2} \pmod p$.

If (a/p) = 1, then a has two square roots (mod p), both of which add up to p.

If (a/p) = -1, then a has no square roots (mod p)

So, if (a/p) = 1, there exist two square roots for a (mod p)

We will see and then discuss here how to compute those square roots x, given a and p.

However note that this algorithm works only when p is prime.

If p = 3 (mod 4), then it is the best case. Square root is given by the following formula

$x = a^{(p+1)/4} \pmod p$

If p = 5 (mod 8), Square root is calculated with the following formula $x = a^{(p+3)/8} \pmod p$

Now, with this, either $x^2 = a \pmod p$ or $x^2 = (p-a) \pmod p$

If $x^2 = (p-a) \pmod p$, then

Assign $x = 2^{(p-1)/4}x \pmod p$

If p = 1 (mod 8), then it is the worst case.

Square root can be calculated by using the following algorithm

Select a random d between 0 and p-1 such that (d/p) = -1, i.e. d is a quadratic non-residue (mod p). Write $p-1 = 2^s t$ where t is odd, i.e. t is the highest odd divisor of p-1

Let $A = a^t \bmod p$

Let $D = d^t \bmod p$

Assign m = 0

For (i from 0 to s-1)

{

If $((AD^m)^{2^{s-1-i}} = -1 \pmod p)$

{

Assign $m = m + 2^i$

}

}

$x = a^{(t+1)/2} D^{m/2} \pmod p$

return x

A.4 SUM OF FACTORS AND EULER'S PHI FUNCTION

The sum of all the positive factors of a given number (including itself) can be found out by the given formula. If the given number m can be written as product of prime factors as $m = p_1^a \times p_2^b \times ... \times p_n^z$, then the sum of factors of m, is given by

$(1 + p_1 + p_1^2 + ... + p_1^a) (1 + p_2 + p_2^2 + ... + p_2^b) ... (1 + p_n + p_n^2 + ... + p_n^z)$

$= (p_1^{a+1}-1)/(p_1-1) \times (p_2^{b+1}-1)/(p_2-1) \times ... \times (p_n^{z+1}-1)/(p_n-1)$

$= ((p_1^{a+1}-1) \times (p_2^{b+1}-1) \times ... \times (p_n^{z+1}-1)) / ((p_1-1) \times (p_2-1) \times ... \times (p_n-1))$

The Euler's Phi Function is defined as the number of numbers, below the given number, and relatively prime to the given number. The property of Phi Function is that if a, m are relatively prime, then $a^{\varphi(m)} \equiv 1 \pmod m$.And if m is prime, then $\varphi(m) = m-1$, So, if p is prime, $a^{p-1} \equiv 1 \pmod p$

The Phi Function can be calculated by using the following formula.

If m is the given number $m = p_1^a \times p_2^b \times ... \times p_n^z$, then $\varphi(m)$ is given by

$m \times (1 - (1/p_1)) \times (1 - (1/p_2)) \times ... \times (1 - (1/p_n)) = (m \times (p_1-1) \times (p_2-1) \times ... \times (p_n-1)) / (p_1 \times p_2 \times ... \times p_n)$

The Möbius function of a given number is defined as 0, if it has at least one repeated prime factor, 1, if all prime factors are distinct and it has even number of prime factors and -1, if all prime factors are distinct and it has odd number of prime factors

A.5 PROPERTIES OF LEGENDRE SYMBOL AND JACOBI SYMBOL

1. $(a/p) = a^{(p-1)/2} \pmod p$
2. $(ab/m) = (a/m).(b/m)$
3. $(a/mn) = (a/m).(a/n)$
4. $(-1/m) = (-1)^{(m-1)/2}$
5. $(2/m) = (-1)^{(m^2-1)/8}$, that is $(2/m) = 1$ if $m \equiv 1$ or $7 \pmod 8$, and $(2/m) = -1$ if $m \equiv 3$ or $5 \pmod 8$

Finally, the law of quadratic reciprocity gives the value of (m/n) if (n/m) is known, and if m and n are relatively prime. If m and n are not relatively prime, then both their value is defined to be 0.

$(m/n).(n/m) = (-1)^{(m-1)(n-1)/4}$, that is if atleast one of m, $n \equiv 1 \pmod 4$, then $(m/n) = (n/m)$. If both of m, $n \equiv 3 \pmod 4$, then $(m/n) = -(n/m)$.

A.6 THE ALIQUOT SEQUENCE

Mersenne Primes are closely associated with Perfect Numbers. Every new discovery of a Mersenne Prime generates a new Perfect Number. A perfect number is a number whose sum of all its factors except itself is equal to the same number. The first few perfect numbers are 6, 28, 496, 8128, 33550336, 8589869056, 137438691328, 2305843008139952128. It is not difficult to prove that if 2^p-1 is prime, then $(2^{p-1})(2^p-1)$ is a perfect number and that every even perfect number has this form. Every even perfect numbers other than 6 will always be congruent to 1 (mod 9), and will end in 6 or 28.

It is not known whether any odd perfect numbers exist, but it has been proved that there aren't any less than 10^{300}. Carl Pomerance has given a heuristic estimate that odd perfect numbers are extremely unlikely. If any odd perfect number exists, then it must be a perfect square times an odd power of a single prime, it must be divisible by atleast 9

distinct primes and has atleast 75 prime factors (not necessarily distinct), and then it has a prime factor greater than 10^{20}.

Let us define the function s(n) to be the sum of all factors of n except itself, and σ(n) to be the sum of all factors of n including itself. To be more clear, s(n) = σ(n) – n. For a perfect number, we have that s(n) = n, or σ(n) = 2n. This function s(n) is called the Restricted Divisor Function. If s(n) > n, then n is called as Abundant Number, and if s(n) < n, then n is known as Deficient Number. For example, 945 is the least Odd Abundant Number, and that any multiple of 6 will always be abundant.

There exist plenty of pair of numbers (x, y) such that s(x) = y and s(y) = x. Such pairs of numbers are called Amicable Numbers.
The amicable pairs below 20000 are given by: (220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368), (10744, 10856), (12285, 14595), (17296, 18416).

Let us iterate the Restricted Divisor Function. Let $s^i(x)$ be the value obtained after i iterations of the Restricted Divisor Function, starting with x. By this process of iterating the Restricted Divisor Function, we obtain a sequence. That sequence is called the Aliquot Sequence. Suppose that $s^k(x) = x$, i.e. after k iterations of the Restricted Divisor Function, starting with x, we again obtain x, then that forms an Aliquot cycle of order k. In order words, x, s(x), $s^2(x)$, $s^3(x)$, …, $s^{k-1}(x)$ are called sociable numbers of order k, where $k \geq 3$.

For example, (12496, 14288, 15472, 14536, 14264) forms an Aliquot cycle of order 5.
The largest known Aliquot cycle is of order 28:
(14316, 19116, 31704, 47616, 83328, 177792, 295488, 629072, 589786, 294896, 358336, 418904, 366556, 274924, 275444, 243760, 376736, 381028, 285778, 152990, 122410, 97946, 48976, 45946, 22976, 22744, 19916, 17716)

There exist plenty of Aliquot cycles of order 4, most of them can be generated by using Thabit rules. The first ten of them being:
(1264460, 1547860, 1727636, 1305184), (2115324, 3317740, 3649556, 2797612), (2784580, 3265940, 3707572, 3370604), (4938136, 5753864, 5504056, 5423384), (7169104, 7538660, 8292568, 7520432), (18048976, 20100368, 18914992, 19252208), (18656380, 20522060, 28630036, 24289964), (28158165, 29902635, 30853845, 29971755), (46722700, 56833172, 53718220, 59090084), (81128632, 91314968, 96389032, 91401368)

We also know five Aliquot cycles of order 6:
(21548919483, 23625285957, 24825443643, 26762383557, 25958284443, 23816997477)
(90632826380, 101889891700, 127527369100, 159713440756, 129092518924, 106246338676)
(1771417411016, 1851936384424, 2118923133656, 2426887897384, 2200652585816, 2024477041144)
(3524434872392, 4483305479608, 4017343956392, 4574630214808, 4018261509992, 3890837171608)
(4773123705616, 5826394399664, 5574013457296, 5454772780208, 5363145542992, 5091331952624)
Two Aliquot cycles of order 8:
(1095447416, 1259477224, 1156962296, 1330251784, 1221976136, 1127671864, 1245926216, 1213138984)
(1276254780, 2299401444, 3071310364, 2303482780, 2629903076, 2209210588, 2223459332, 1697298124)
and one Aliquot cycle of order 9:
(805984760, 1268997640, 1803863720, 2308845400, 3059220620, 3367978564, 2525983930, 2301481286, 1611969514)

That is about Aliquot cycles. Coming back to Aliquot Sequences, let us take some examples starting with arbitrary values.

12, 16, 15, 9, 4, 3

30, 42, 54, 66, 78, 90, 144, 259, 45, 33, 15, 9, 4, 3

60, 108, 172, 136, 134, 70, 74, 40, 50, 43

96, 156, 236, 184, 176, 196, 203, 37

120, 240, 504, 1056, 1968, 3240, 7650, 14112, 32571, 27333, 12161

48, 76, 64, 63, 41

24, 36, 55, 17

18, 21, 11

Let us now consider with a longer sequence

180, 366, 378, 582, 594, 846, 1026, 1374, 1386, 2358, 2790, 4698, 6192, 11540, 12736, 12664, 11096, 11104, 10820, 11944, 10466, 5236, 6860, 9940, 14252, 14308, 15218, 10894, 6746, 3376, 3196, 2852, 2524, 1900, 2440, 3140, 3496, 3704, 3256, 3584, 4600, 6560, 9316, 8072, 7078, 3542, 3370, 2714, 1606, 1058, 601

Another example

11025, 11946, 14262, 14274, 19578, 22758, 22770, 44622, 56154, 75174, 101082, 113190, 232410, 338982, 450354, 470094, 490674, 509838, 680562, 844764, 1314372, 1952108, 1496764, 1132100, 1324774, 843074, 428734, 228194, 119134, 59570, 71758, 35882, 31510, 28106, 20278, 10142, 6490, 6470, 5194, 4040, 5140, 5696, 5734, 3194, 1600, 2337, 1023, 513, 287, 49, 8, 7

All these sequences terminate in a prime. Some sequences can terminate in a perfect number, or an amicable pair, or Aliquot cycle of higher order as well. For example:

2195, 445, 95, 25, 6, 6…

650, 652, 496, 496…

2856, 5784, 8736, 19488, 40992, 84000, 230496, 475356, 792484, 1013852, 1013908, 1058092, 1264340, 2049964, 2123576, 2778664, 3492536, 3077104, 2884816, 3391568, 3775384, 3303476, 3003244, 2288756, 1773484, 1558964, 1440718, 720362, 360184, **376736**, 381028, 285778, 152990, 122410, 97946, 48976, 45946, 22976, 22744, 19916, 17716, 14316, 19116, 31704, 47616, 83328, 177792, 295488, 629072, 589786, 294896, 358336, 418904, 366556, 274924, 275444, 243760, **376736**, …

Now try that out with some longer sequences as well:

$s^{176}(138) = 59$, $s^{299}(702) = 191$, $s^{193}(720) = 277$, $s^{183}(936) = 43$, $s^{800}(4488) = 43$, $s^{671}(1848) = 43$, $s^{177}(976950) = 6$, $s^{99}(980460) = 2620$, $s^{228}(17490) = 1264460$, $s^{393}(2880) = 277$, $s^{746}(840) = 601$, $s^{1074}(1248) = 37$, $s^{166}(858) = 59$

In 1888, Eugène Catalan conjectured that all Aliquot Sequences terminate in a prime, or perfect number or amicable pair or Aliquot cycle of higher order. This is one of the greatest problems of computational number theory. It is unlikely that this conjecture is true, for there are some sequences that grow indefinitely without bound, without even catching up in a cycle. These sequences are called open-end sequences. According to the current factoring resources, these sequences have been extended atleast upto 120 digits in size.

Today, there are twelve open-end sequences below 1000:

276, 306, 396, 552, 564, 660, 696, 780, 828, 888, 966, 996

But, notice that (276, 306, 396, 696) all belong to the same family. And so does (660, 828, 996), (564, 780) and (552, 888)

Considering the smallest member of distinct sequences, we have 5 open-end sequences below 1000, namely 276, 552, 564, 660, 966. These are called as “Lehmer Five”. It was originally known as “Lehmer Six” along with 840, which had been later on terminated by several groups separately.

The open-end sequences between 1000 and 2000 were termed as “Godwin Fourteen” including 1248 and 1848, but today there are only twelve of them: 1074, 1134, 1464, 1476, 1488, 1512, 1560, 1578, 1632, 1734, 1920, 1992.

Today, there are exactly 81 open-end sequences below 10000.

276, 552, 564, 660, 966, 1074, 1134, 1464, 1476, 1488, 1512, 1560, 1578, 1632, 1734, 1920, 1992, 2232, 2340, 2360, 2484, 2514, 2664, 2712, 2982, 3270, 3366, 3408, 3432, 3564, 3678, 3774, 3876, 3906, 4116, 4224, 4290, 4350, 4380, 4788, 4800, 4842, 5148, 5208, 5250, 5352, 5400, 5448, 5736, 5748, 5778, 6396, 6552, 6680, 6822, 6832, 6984, 7044, 7392, 7560, 7890, 7920, 8040, 8154, 8184, 8288, 8352, 8760, 8844, 8904, 9120, 9282, 9336, 9378, 9436, 9462, 9480, 9588, 9684, 9708, 9852

Any progress in computations can certainly be able to reduce these numbers down.

A.7 WIEFERICH PRIMES

Any prime number p satisfies with $a^{p-1} \equiv 1 \pmod{p}$ for any base 'a' such that GCD(a, p) = 1. Thus, $2^{p-1} \equiv 1 \pmod{p}$ for all primes $p \geq 3$. A Wieferich prime is a prime p which satisfies with $2^{p-1} \equiv 1 \pmod{p^2}$. If square of some prime q divides a Mersenne number with a prime exponent p, i.e. $q^2 \mid 2^p-1$, where p and q are primes, then q must be a Wieferich prime. The Wieferich prime concept was first introduced by Wieferich in 1909, that if the first case of Fermat's Last Theorem is false for prime p, then p must be a Wieferich prime. Only two Wieferich primes are known till today, 1093 and 3511, found out by Meissner in 1913 and Beeger in 1922 respectively. This search has been carried out exhaustively upto 6.7 quadrillion.

To answer whether there exist any prime q such that $q^2 \mid 2^p-1$, where p is also prime. 1093 does not satisfy the necessary condition, i.e. $2^{(p-1)/2} \equiv 1 \pmod{p^2}$, that is also needed for the above statement to be true. 3511 does not divide any Mersenne number with prime exponent at all, so no such prime q is being known till today.

Similar to Wieferich primes, that satisfy with $2^{p-1} \equiv 1 \pmod{p^2}$, that are based upon the Fermat's Little Theorem $a^{p-1} \equiv 1 \pmod{p}$. There is also some research carried upon Wilson primes, that satisfy with $(p-1)! + 1 \equiv 0 \pmod{p^2}$, that is based upon the Wilson's Theorem $(p-1)! + 1 \equiv 0 \pmod{p}$. And then Wall-Sun-Sun primes, that satisfy with $F_{p-(p/5)} \equiv 0 \pmod{p^2}$, that is based upon the Fibonacci number primality test, that is $F_{p-(p/5)} \equiv 0 \pmod{p}$. Here F_k denotes the k^{th} Fibonacci number, and that (p/5) is the Legendre Symbol. The only known Wilson primes are 5, 13 and 563 (below 500 million), and then no Wall-Sun-Sun prime is being known till today (below 200 trillion).

A.8 FACTORS OF SOME SPECIAL NUMBERS

Some people believe that if $q = 2^p-1$ is prime, then 2^q-1 is also prime. However, that is not true. But if $q = 2^p-1$ is composite (with some prime factor F), then 2^q-1 will always be composite (with 2^F-1 as a factor). For the first four Mersenne primes 2, 3, 5 and 7,

however 2^3-1, 2^7-1, 2^{31}-1 and 2^{127}-1 are all prime. But for the next four Mersenne primes 13, 17, 19 and 31, we know that 2^{8191}-1, 2^{131071}-1, 2^{524287}-1, $2^{2147483647}$-1 all are composite.

2^{8191}-1 has factors 338193759479, 210206826754181103207028761697008013415622289

2^{131071}-1 has factors 231733529, 64296354767

2^{524287}-1 has factors 62914441, 5746991873407, 2106734551102073202633922471, 824271579602877114508714150039

$2^{2147483647}$-1 has factors 295257526626031, 87054709261955177, 242557615644693265201, 178021379228511215367151

Tony Forbes is leading a project to search for a factor of the next term $2^{2305843009213693951}$-1, none being found out so far.

Coming to the case of Fermat primes, 2^3-1, 2^5-1, 2^{17}-1 all are primes. But the next term 2^{257}-1 is not prime. 2^{257}-1 = 535006138814359 × 1155695385246619182673033 × 374550598501810936581776630096313181393

2^{65537}-1 has factors 513668017883326358119, 88385234056578716 4089923172087

The first few Mersenne numbers, that are known to be composite, but with no known prime factors at all, till today are 2^{1061}-1, 2^{1277}-1, 2^{1619}-1, 2^{1753}-1.

A.9 SUM OF SQUARES

Sum of two squares:

Every prime number of the form 1 (mod 4) can be uniquely represented as sum of two squares. For example, $5 = 2^2 + 1^2$, $13 = 3^2 + 2^2$, $17 = 4^2 + 1^2$, $29 = 5^2 + 2^2$, $37 = 6^2 + 1^2$, $41 = 5^2 + 4^2$, $53 = 7^2 + 2^2$, $61 = 6^2 + 5^2$, $73 = 8^2 + 3^2$, $89 = 8^2 + 5^2$, $97 = 9^2 + 4^2$.

2 has also unique representation: $1^2 + 1^2$. Any number of form 3 (mod 4) cannot be represented as sum of two squares in any way. This algorithm can be used to return the values (a, b) for a^2+b^2 representation for any arbitrary prime number p of form 1 (mod 4).

Initialize a = p, choose any quadratic non-residue (mod p) and then assign its value to variable x. Initialize $b = x^{(p-1)/4} \pmod p$.

While ($b^2 > p$)

{

(a, b) = (b, a mod b)

}

Return (b, a mod b)

If any integer N has a prime factor of form 3 (mod 4) to an odd power, then N cannot be represented as sum of two squares. If N has all prime factors of form 3 (mod 4) to even powers (≥2), then N has no sum of two squares representation in lowest terms.
That is, if $N = y^{2k}z$, where y^{2k} is product of all prime factors of N of form 3 (mod 4), and that z is product of all prime factors of N of form 1 (mod 4), then for every sum of two squares representation of $z = q^2 + r^2$, the only possible representations of N as sum of two squares are $N = (y^k q)^2 + (y^k r)^2$.

If N is the product of two distinct primes of form 1 (mod 4), then N will have two different representations as sum of two squares. This is demonstrated by using the following equation:
$(a^2 + b^2)(x^2 + y^2) = |ax + by|^2 + |ay - bx|^2 = |ax - by|^2 + |ay + bx|^2$
This can be similarly extended when N is product of multiple primes of form 1 (mod 4) not necessarily distinct. If N is the product of k distinct primes of form 1 (mod 4), then N can be represented as sum of two squares in 2^{k-1} ways. If the prime factors of N are repeated, then there will be duplicate entries by using the above formula, so the number of possible ways of representing N as sum of two squares will be lesser than that.

If N has certain number of ways to be represented as sum of two squares, then 2N can also be represented as sum of two squares in the same number of ways. It becomes evident from the following equation: $2(a^2 + b^2) = (a + b)^2 + (a - b)^2$. Substituting with x = 1 and y = 1 in the above equation, we simply arrive at this equation.

Sum of three squares: Every positive integer other than those of form $4^b(8c + 7)$, $b \geq 0$, $c \geq 0$ can be represented as sum of three squares of non-negative integers (including zero).

Sum of four squares: Every positive integer can be represented as sum of four squares of non-negative integers (including zero). The Euler's identity clears this assertion.
$(a^2 + b^2 + c^2 + d^2)(e^2 + f^2 + g^2 + h^2) = (ae + bf + cg + dh)^2 + (af - be + ch - dg)^2 + (ag - ce + bh - df)^2 + (ah - de + bg - cf)^2$

A.10 COMPUTING MINIMAL EQUAL SUMS OF LIKE POWERS

The Fermat's Last Theorem had been a long-standing unsolved problem in Mathematics for over 300 years until it was resolved by Andrew Wiles in 1995, by using topics in very advanced mathematics, elliptic curves and modular forms. It states that the Diophantine equation $x^n + y^n = z^n$ has no solution in positive integers for any value of $n \geq 3$. Fermat himself proved it for $n = 4$, by using method of infinite descent. Euler proved for $n = 3$, and then Kummer proved it for all regular primes.

When $n = 2$, then the equation $x^2 + y^2 = z^2$ becomes the Pythagoreas Equation, which has infinitely many solutions in lowest terms given by $((m^2 - n^2)/2, 2mn, (m^2 + n^2)/2)$, where m, n are odd, with $m > n$, gcd (m, n) = 1. This is the general solution for the above Pythagoreas Equation for that (x, y, z) triplet in the lowest terms.

Beal's Conjecture: Analogous to the Fermat's Last Theorem, it has been conjectured that the Diophantine equation $x^a + y^b = z^c$ has no solution in positive integers for (x, y, z) when all of $(a, b, c) \geq 3$, when gcd (x, y, z) = 1.

If (x, y, z) are not co-prime to each other, then there exist solutions to the above equation such as $756^3 + 945^3 = 189^4$.

There also exist some solutions to the above equation when not all of $(a, b, c) \geq 3$, like $17^3 + 2^7 = 71^2$, $3^5 + 10^2 = 7^3$, $11^2 + 2^2 = 5^3$, $7^2 + 2^5 = 3^4$, $3^5 + 11^4 = 122^2$, $13^2 + 7^3 = 2^9$, $1^7 + 3^2 = 2^3$, etc. All these equations involve atleast some square term within them, for example, $33^8 + 1549034^2 = 15613^3$, $1414^3 + 2213459^2 = 65^7$, $9262^3 + 15312283^2 = 113^7$, $17^7 + 76271^3 = 21063928^2$, $43^8 + 96222^3 = 30042907^2$.

Computing Minimal Equal Sums of Like Powers: Let us now discuss about writing sum of M k^{th} powers as sum of N different k^{th} power terms.

From the Fermat's Last Theorem, it follows that $x^3 + y^3 = z^3$ has no solution in positive integers. But for the equation $x^3 + y^3 = z^3 + w^3$ has many solutions.

$10^3 + 9^3 = 12^3 + 1^3$ is Ramanujan's Taxicab Number. Also that $16^3 + 2^3 = 15^3 + 9^3$.

$x^3 + y^3 + z^3 = w^3$ also has got some solutions, too. For example, $3^3 + 4^3 + 5^3 = 6^3$.

That is for cubes. Coming to the fourth powers, $x^4 + y^4 = z^4$ has obviously got no solution in positive integers. But that $x^4 + y^4 = z^4 + w^4$ as well as $x^4 + y^4 + z^4 = w^4$ both have got atleast some solutions. $158^4 + 59^4 = 134^4 + 133^4$. $422481^4 = 414560^4 + 217519^4 + 95800^4$.

For the higher powers, I have got some examples with me as well.

$144^5 = 133^5+110^5+84^5+27^5$

$14132^5+220^5 = 14068^5+6237^5+5027^5$

$23^6+15^6+10^6 = 22^6+19^6+3^6$

$966^8+539^8+81^8 = 954^8+725^8+481^8+310^8+158^8$

$3113^8+2012^8+1953^8+861^8=2823^8+2767^8+2557^8+1128^8$

Thus, it is conjectured that for forming such an equation of degree k, we need atleast k terms.

A.11 SOME SPECIAL DIOPHANTINE EQUATIONS

A Diophantine Equation always seeks up for integer values as solutions, or that lattice points within the Cartesian co-ordinate system.

The Diophantine equation $2^n - 7 = m^2$ has got no solutions other than that for (m, n) = (1, 3), (3, 4), (5, 5), (11, 7), (181, 15).

This is closely related with Triangular Numbers of form $2^n - 1$, since they are given by the equation $2^n - 1 = m(m-1)/2$, or that $2^{n+3} - 7 = (2m - 1)^2$.

Also, notice that whether the equation $n! + 1 = m^2$ has got any other solutions besides that for (m, n) = (5, 4), (11, 5), (71, 7).

The linear Diophantine equation is given by ax+by = c, and that equation has got solutions for x, y if and only if c is a multiple of gcd(a, b).

Next, take up with that quadratic Diophantine equation, that is the Pell's Equation.
This equation $|x^2 - 2y^2| = 1$, has got solutions for x, y that are given by using the convergents of the continued fraction expansion for √2. (x, y) is a solution for that above equation if and only if x/y is a convergent of the continued fraction expansion of √2.
For example, the first few solutions are given as: (1, 1), (3, 2), (7, 5), (17, 12), (41, 29), (99, 70)… Note that $x_i = 2x_{i-1} + x_{i-2}$, and then that $y_i = 2y_{i-1} + y_{i-2}$ as well. Also note that the value of $x^2 - 2y^2$ alternates between 1 and -1 for each of these adjacent solutions.
In case it is not clear, 1/1, 3/2, 7/5, 17/12, 41/29, 99/70 are all convergents of continued fraction expansion of √2, because of the fact that it is an approximation of value (as a fraction) for this series, after every fixed number of terms within that: 1+1/(2+1/(2+1/(2+1/…)))

A.12 STRAIGHTEDGE AND COMPASS CONSTRUCTIONS

A straightedge is a ruler with no markings on it. Given a unit length, what lengths relative to it can you construct by using straightedge and compass? What regular polygons can you construct by using the given straightedge and compass? Let us answer that question over here.

Given a straightedge and compass and a piece of paper, you are allowed to
Mark a random point on paper.
Draw a line, line segment or ray between two fixed points.
Draw a circle with center as one point, and passing through another existing point.
Mark intersection of two circles, two lines or line and circle.
Hide any unwanted points, lines or circles.

Marking middle of any two given points, drawing perpendicular bisector of any given line segment, bisection of any given angle are possible with these facilities.

We can construct lengths of square roots of any integer. For example, $\sqrt{2}$ is the length of hypotenuse of right angled triangle whose other two sides are 1 and 1, $\sqrt{3}$ is the length of hypotenuse of right angled triangle whose other two sides are $\sqrt{2}$ and 1, and so on.

We can divide a line segment into any number of equal parts. Suppose that you want to divide it into N equal parts, then take another parallel line segment containing N+1 points that are equally spaced, and then project that original line segment with this parallel line, with the end points of the original line segment corresponding to that of the end points of the parallel line segment.

Regular polygons: It is quite trivial to construct a polygon of 2N sides from a polygon of N sides by bisecting the angle between the center of polygon to its adjacent vertices or by joining the center of polygon to the mid-points of each of the edges of the polygon of N sides and the points where these lines meet the circumscribed circle of the polygon of N sides correspond to the other vertices of polygon of 2N sides.

To construct any regular polygon of N sides, you know the center of polygon and one of its vertices, and you are required to construct the other vertices and then edges of the polygon.

An equilateral triangle can be easily constructed by using straightedge and compass by using the fact that its altitude is $\sqrt{3}/2$ times its side. Note that the altitude also bisects one of its sides.

Construction of a square needs no explanation at all. It is very obvious enough.

A regular pentagon is not hard enough to construct. Every internal angle is 108°, and we have that $\cos 72° = (\sqrt{5} - 1)/4$. Every angle between the center of pentagon to its adjacent vertices is 72°, and we also have that the diagonal of the pentagon is $p_1 = \sqrt{(5-\sqrt{5})/2}$ times the radius of its circumscribed circle. In order to realize this, first of all we need to construct a length of $p_2 = (\sqrt{5} - 1)/2$, and then construct the length of $p_1 = \sqrt{(p_2^2 + 1)}$, by using the Pythagoreas Theorem.

Regular hexagon can be constructed from equilateral triangle, regular octagon from square, and so on.

A regular polygon of 15 sides can be constructed by using an equilateral triangle and then a regular pentagon. Place all their vertices over a circle, with atleast one vertex of equilateral triangle and regular pentagon being coincident. Then, there exist atleast one vertex of the equilateral triangle and one vertex of regular pentagon that forms the adjacent vertices of the regular polygon of 15 sides.

This trick cannot be applied to construct a regular polygon of 18 sides from an equilateral triangle and regular hexagon, because of the fact that 3 and 6 are not co-prime.

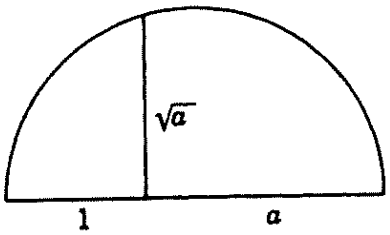
In 1796, at the age of 19, Gauss realized that it is possible to construct a regular polygon of 17 sides by using only straightedge and compass. A regular polygon of N sides can be constructed by using straightedge and compass if and only if N is a power of 2 times 4 or product of one or more distinct Fermat primes. The only known Fermat primes are 3, 5, 17, 257 and 65537. Then, it is known that there exist no more Fermat primes at all till F_{32}.

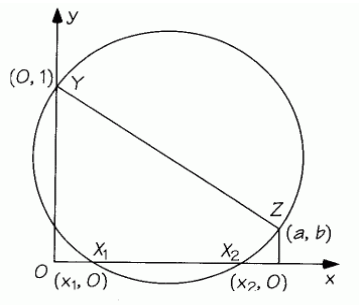
A length X relative to a given unit length can be constructed by using straightedge and compass if and only if X is an element of quadratic extension field of rational numbers. It means that X should be a root of an irreducible polynomial equation with rational coefficients whose degree is a power of 2. For example, $\cos(\pi/7)$ is a root of irreducible cubic $8x^3+4x^2-4x-1 = 0$, and that $\cos(\pi/9)$ is a root of irreducible cubic $8x^3-6x+1 = 0$. These lengths cannot be constructed because the degree of irreducible polynomial is 3, and not a power of 2. Thus, it becomes evident that regular heptagon or regular nonagon cannot be constructed by using straightedge and compass only. Note that there also exist certain irrational numbers that do not satisfy any polynomial equation with rational coefficients at all. These are called transcendental numbers. π and e are examples of transcendental numbers. Other irrational numbers are said to be algebraic. Lengths of transcendental numbers cannot be also constructed by using straightedge and compass only.

We can construct a rational length times any given length, because you can make any number of copies of given length, concatenate it, and then divide that line segment into any number of equal parts.

Square root of any given length X is also possible. Take a line segment of length X+1, and then draw a semicircular arc with this line segment as diameter. Then, draw a perpendicular at a distance of 1 from the end point of its line segment. Then, the height at which this perpendicular cuts the semicircular arc will be equal to $\sqrt{X}$. In this way, we can construct lengths like $\sqrt[4]{2}$ by using only straightedge and compass, that is the square root of $\sqrt{2}$ itself. (Figure to the left)

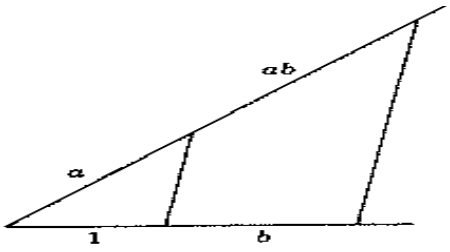
Suppose that you want to construct the lengths that are roots of quadratic equation $x^2 - ax + b = 0$, then draw a circle with coordinates (a, b) and then (0, 1) as the end points of diameter of the circle. Then, the X-coordinates of the points where this circle cuts up with the X-axis will give away with the lengths that are the roots of this quadratic equation. (Figure to the right)

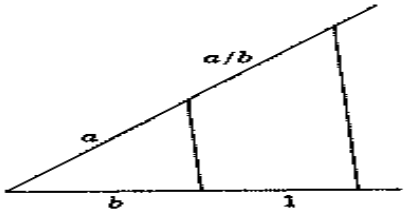




If we take up with a line segment with two parts of lengths 1 and b, and then project the first part with length a, then the length of projection of second part will be ab (Multiplication of two lengths)

If we take up with a line segment with two parts of lengths b and 1, and then project the first part with length a, then the length of projection of second part will be a/b (Division of two lengths) – (Illustrations being presented upon the next page).





Constructions that cannot be done with the aid of straightedge and compass only are:
Squaring a circle – Constructing a square with the equal area of any given circle. Here we assume that circle of radius unit length. This involves construction of side of square with length $\sqrt{\pi}$. It is not possible since $\sqrt{\pi}$ is a transcendental number.
Doubling a cube – Given a cube of unit length, construct a cube whose volume is twice that of the original cube. This is not possible, since the length of edge of this new cube is $\sqrt[3]{2}$, which is a root of irreducible cubic $x^3 - 2 = 0$, with rational coefficients is of degree 3, that is not a power of 2 at all.
Trisection of an arbitrary angle is not possible by using only straightedge and compass as well.

Some constructions that are possible by using only straightedge and compass include:
Tangent to circle from a point on circle, Tangents to circle from a point outside circle, Trisection of right angle, A line that is parallel to any given line, perpendicular to a line from a point on the given line, perpendicular to a line from a point outside the given line, Angle bisectors, partitioning a given line segment into any number of equal parts, centroid, orthocentre, circumcentre, circumcircle, incentre, incircle, excentre, excircle of any given triangle, center of any given circle, diameter of circle, middle of two points, perpendicular bisector of any given line segment, parallelogram, square with any given side, square with any given diagonal, Translation of any given length on a line to any other part over the same given line itself.

Some constructions that I am not sure of: Direct and inverse common tangents to any given two circles, Ten cases of Appolonius problem, that is construction of circles that are being tangential to any three given elements, where that given element may be either a point, a line or an other circle itself, as well.

APPENDIX B

NUMERICAL RESULTS

B.1 AURIFEUILLIAN FACTORIZATIONS

$(2^{4k-2}+1) = (2^{2k-1}-2^k+1)(2^{2k-1}+2^k+1)$

$(3^{6k-3}+1) = (3^{2k-1}+1)(3^{2k-1}-3^k+1)(3^{2k-1}+3^k+1)$

$(5^{10k-5}-1) = (5^{2k-1}-1)(5^{4k-2}-5^{3k-1}+3.5^{2k-1}-5^k+1)(5^{4k-2}+5^{3k-1}+3.5^{2k-1}+5^k+1)$

$(6^{12k-6}+1) = (6^{4k-2}+1)(6^{4k-2}-6^{3k-1}+3.6^{2k-1}-6^k+1)(6^{4k-2}+6^{3k-1}+3.6^{2k-1}+6^k+1)$

$(7^{14k-7}+1) = (7^{2k-1}+1)(7^{6k-3}-7^{5k-2}+3.7^{4k-2}-7^{3k-1}+3.7^{2k-1}-7^k+1)(7^{6k-3}+7^{5k-2}+3.7^{4k-2}+7^{3k-1}+3.7^{2k-1}+7^k+1)$

$(10^{20k-10}+1) = (10^{4k-2}+1)(10^{8k-4}-10^{7k-3}+5.10^{6k-3}-2.10^{5k-2}+7.10^{4k-2}-2.10^{3k-1}+5.10^{2k-1}-10^k+1)(10^{8k-4}+10^{7k-3}+5.10^{6k-3}+2.10^{5k-2}+7.10^{4k-2}+2.10^{3k-1}+5.10^{2k-1}+10^k+1)$

$(11^{22k-11}+1) = (11^{2k-1}+1)(11^{10k-5}-11^{9k-4}+5.11^{8k-4}-11^{7k-3}-11^{6k-3}+11^{5k-2}-11^{4k-2}-11^{3k-1}+5.11^{2k-1}-11^k+1)(11^{10k-5}+11^{9k-4}+5.11^{8k-4}+11^{7k-3}-11^{6k-3}-11^{5k-2}-11^{4k-2}+11^{3k-1}+5.11^{2k-1}+11^k+1)$

$(12^{6k-3}+1) = (12^{2k-1}+1)(12^{2k-1}-2^{2k-1}.3^k+1)(12^{2k-1}+2^{2k-1}.3^k+1)$

B.2 THE FIRST 33 FERMAT NUMBERS $F_n = 2^{2^n}+1$

$F_0 = 3$

$F_1 = 5$

$F_2 = 17$

$F_3 = 257$

$F_4 = 65537$

$F_5 = 641 \times 6700417$

$F_6 = 274177 \times 67280421310721$

$F_7 = 59649589127497217 \times 5704689200685129054721$

$F_8 = 1238926361552897 \times$
$93461639715357977769163558199606896584051237541638188580280321$

$F_9 = 2424833 \times 7455602825647884208337395736200454918783366342657 \times 741640062627530801524787141901937474059940781097519023905821316144415759$
$5047050080928187116939407 37$

$F_{10} = 45592577 \times 6487031809 \times 4659775785220018543264560743076778192897 \times P_{252}$

$F_{11} = 319489 \times 974849 \times 167988556341760475137 \times 3560841906445833920513 \times P_{564}$

$F_{12} = 114689 \times 26017793 \times 63766529 \times 190274191361 \times 1256132134125569 \times 568630647535356955169033410940867804839360742060818433 \times C_{1133}$

$F_{13} = 2710954639361 \times 2663848877152141313 \times 3603109844542291969 \times 319546020820551643220672513 \times C_{2391}$

$F_{14} = 116928085873074369829035993834596371340386703423373313 \times C_{4880}$

$F_{15} = 1214251009 \times 2327042503868417 \times 168768817029516972383024127016961 \times C_{9808}$

$F_{16} = 825753601 \times 188981757975021318420037633 \times C_{19694}$

$F_{17} = 31065037602817 \times C_{39444}$

$F_{18} = 13631489 \times 81274690703860512587777 \times C_{78884}$

$F_{19} = 70525124609 \times 646730219521 \times 375900555141337542865244460804997 13 \times C_{157770}$

$F_{20} = C_{315653}$

$F_{21} = 4485296422913 \times C_{631294}$

$F_{22} = 64658705994591851009055774868504577 \times C_{1262577}$

$F_{23} = 167772161 \times C_{2525215}$

$F_{24} = C_{5050446}$

F_{25} has factors 25991531462657, 204393464266227713, 2170072644496392193

F_{26} has a factor 76861124116481

F_{27} has factors 151413703311361, 231292694251438081

F_{28} has a factor 1766730974551267606529

F_{29} has a factor 2405286912458753

F_{30} has factors 640126220763137, 1095981164658689

F_{31} has a factor 46931635677864055013377

F_{32} has a factor 25409026523137

B.3 FACTORS OF MERSENNE NUMBERS

(2^n-1, with n < 150 and n is prime)

n	Prime Factorization of 2^n-1
2	3
3	7
5	31
7	127
11	23.89
13	8191
17	131071
19	524287
23	47.178481
29	233.1103.2089
31	2147483647
37	223.616318177
41	13367.164511353
43	431.9719.2099863
47	2351.4513.13264529
53	6361.69431.20394401
59	179951.3203431780337
61	2305843009213693951
67	193707721.761838257287
71	228479.48544121.212885833
73	439.2298041.9361973132609
79	2687.202029703.1113491139767
83	167.57912614113275649087721
89	618970019642690137449562111
97	11447.13842607235828485645766393
101	7432339208719.341117531003194129
103	2550183799.3976656429941438590393

107 162259276829213363391578010288127

109 745988807.87003598609872098733287 3

113 3391.23279.65993.1868569.1066818132868207

127 170141183460469231731687303715884105727

131 263.1035079443105516238671861923746823456 9

137 32032215596496435569.5439042183600204290159

139 5625767248687.123876132205208335762278423601

149 866562685662821831 51.8235109336690846723986161

B.4 LIST OF MERSENNE, WAGSTAFF, REPUNIT PRIMES

Mersenne Primes are values of n such that $2^n - 1$ is prime

2 3 5 7 13 17 19 31 61 89 107 127 521 607 1279 2203 2281 3217 4253 4423

9689 9941 11213 19937 21701 23209 44497 86243 110503 132049 216091

756839 859433 1257787 1398269 2976221 3021377 6972593 13466917

20996011 24036583 25964951 30402457 32582657 37156667 42643801 43112609

Wagstaff Primes are values of n such that $(2^n + 1)/3$ is prime

2 3 5 7 11 13 17 19 23 31 43 61 79 101 127 167 191 199 313 347 701 1709 2617

3539 5807 10501 10691 11279 12391 14479 42737 83339 95369 117239 127031

138937 141079 267017 269987 374321 986191 4031399

Repunit Primes are values of n such that $(10^n - 1)/9$ is prime

2 19 23 317 1031 49081 86453 109297 270343